NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A SPLIT-LEVINSON APPROACH TO AUTOREGRESSIVE MODELING

by

William A. Dicken

June 1988

Thesis Advisor Murali Tummala

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

1a Report Security Classification: Unclassified		1b Restrictive Markings	
2a Security Classification Authority		3 Distribution Availability of Report: Approved for public release: distribution is unlimited.	
2b Declassification/Downgrading Schedule			
4 Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)	
6a Name of Performing Organization: Naval Postgraduate School	6b Office Symbol (if applicable): 32	7a Name of Monitoring Organization: Naval Postgraduate School	
6c Address (city, state, and ZIP code): Monterey, CA 93943-5000		7b Address (city, state, and ZIP code): Monterey, CA 93943-5000	
8a Name of Funding/Sponsoring Organization	8b Office Symbol (if applicable)	9 Procurement Instrument Identification Number	
8c Address (city, state, and ZIP code)		10 Source of Funding Numbers	
		Program Element No	Project No / Task No / Work Unit Accession No

11 Title (include security classification): A SPLIT-LEVINSON APPROACH TO AUTOREGRESSIVE MODELING

12 Personal Author(s): William A. Dicken

13a Type of Report	13b Time Covered	14 Date of Report (year, month, day)	15 Page Count
Master's Thesis	From To	June 1988	76

16 Supplementary Notation: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms (continue on reverse if necessary and identify by block number)
Field	Group	Subgroup	thesis, word processing, Script, GML, text processing.

19 Abstract (continue on reverse if necessary and identify by block number): The classical Levinson-Durbin linear prediction formulas for real valued input sequences are examined and compared to the recently proposed split-Levinson formulas. Both the autoregressive linear predictor model and the adaptive lattice model are used to formulate the new split-Levinson algorithms. A brief introduction to the theory of symmetric polynomials is presented to form the basis of the new algorithms. Computer simulations are used to test and compare the computational accuracy of the new algorithms for AR filter coefficient estimation, parameter estimation for a moving average process, and spectral estimation of sinusoids in white noise. Research results indicate that the new algorithms reduce the number of real multiplications required for a k^{th} order AR filter problem by one-half, and they are applicable to both the extended Prony method of spectral estimation and the estimation of moving average parameters.

20 Distribution Availability of Abstract	21 Abstract Security Classification	
☒ unclassified unlimited ☐ same as report ☐ DTIC users	Unclassified	
22a Name of Responsible Individual: Murali Tummala	22b Telephone (include Area code): (408) 646-2645	22c Office Symbol: 62Tu

DD FORM 1473,84 MAR — 83 APR edition may be used until exhausted; All other editions are obsolete

Approved for public release; distribution is unlimited.

A SPLIT-LEVINSON APPROACH TO AUTOREGRESSIVE MODELING

by

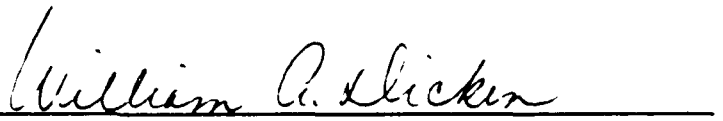
William A. Dicken
Lieutenant Commander, United States Navy
B.S., University of Mississippi, 1977

Submitted in partial fulfillment of the
requirements for the degree of

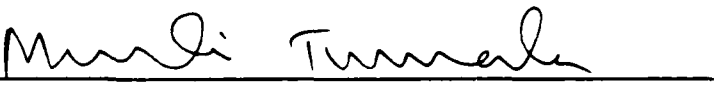
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

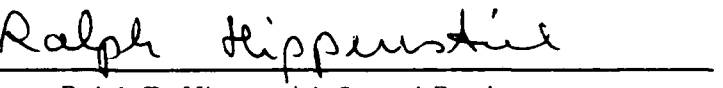
NAVAL POSTGRADUATE SCHOOL
June 1988

Author: 

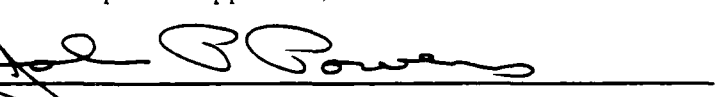
William A. Dicken

Approved by: 

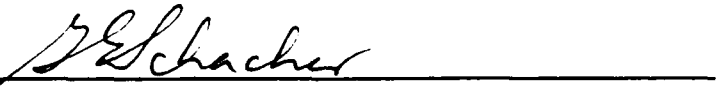
Murali Tummala, Thesis Advisor



Ralph D. Hippenstiel, Second Reader



John P. Powers, Chairman,
Department of Electrical Engineering



Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

The classical Levinson-Durbin linear prediction formulas for real valued input sequences are examined and compared to the recently proposed split-Levinson formulas. Both the autoregressive linear predictor model and the adaptive lattice model are used to formulate the new split-Levinson algorithms. A brief introduction to the theory of symmetric polynomials is presented to form the basis of the new algorithms. Computer simulations are used to test and compare the computational accuracy of the new algorithms for AR filter coefficient estimation, parameter estimation for a moving average process, and spectral estimation of sinusoids in white noise. Research results indicate that the new algorithms reduce the number of real multiplications required for a k^{th} order AR filter problem by one-half, and they are applicable to both the extended Prony method of spectral estimation and the estimation of moving average parameters.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. OBJECTIVE

The classical Levinson algorithm is known to provide solutions to real valued, linear systems involving Toeplitz structures. The computational cost for these solutions, is known to be $O(k^2)$, where k indicates the filter order. It has recently been proposed that the classical algorithm may be transformed into 2 simpler algorithms, using the theory of symmetric polynomials, and that either of these algorithms can be used to to solve for the predictor polynomial of order k at a reduced computational cost. [Ref. 1: p. 470]

These new algorithms are termed the split-Levinson algorithms because their basis is formed from the concept of symmetric polynomials. These are not new in theory, but the application of the process to linear prediction is a new concept. Symmetric polynomials are based on the Barlett Bisection Theorem [Ref. 2 : pp. 1074-1076], where a system that possesses symmetry about a point, such as a Toeplitz matrix, can be decomposed into a symmetric and an antisymmetric part. The unique point of the theory is that either part may be used to solve the problem, or a combination of both parts can also be used in the solution. During our research we shall only consider real data sequences.

The split-Levinson case also has a lattice structure as the classical case. However, as will be shown, the structure of this lattice shows little resemblance to its classical counterpart. A derivation of a revised split lattice structure, and its recursive algorithm was attempted in order to represent the split lattice in a form similar to that of the classical structure. Although unsuccessful, the derivation procedure is presented for subject matter continuity.

Computer programs have been written to implement the new algorithms and compare them to the classical algorithms. Additionally, computer programs are included to apply the split-Levinson algorithm for two cases, where the computational efficiency of the new algorithm could be of substantial benefit. These cases include the Moving Average (MA) problem, where the parameters of a MA model must be determined from the given data, and an extension of the Prony method of spectral estimation, where a least squares estimation of the presence of sinusoids in white noise is made from the output data sequence.

This thesis compares the classical and lattice structures of the Levinson recursion formula given in [Ref. 3: pp. 145-167], and examines not only the formulation of the recursion formulas for these algorithms, but also the complexity of the computations and the resulting structure of each of the algorithms.

B. THESIS ORGANIZATION

The structure of the thesis is divided into 4 chapters, including the Introduction. In Chapter II we will review the classical Levinson algorithms. In the first case, the algorithm is obtained using the autocorrelation function of the input sequence, and in the second case, it is obtained using the forward and backward error vectors of the input sequence. In each case we shall establish the number of real multiplications required to complete a k-th order recursion of the respective algorithm. As stated, the ultimate goal is to establish the computational efficiency of the split-Levinson algorithm over the classical Levinson algorithm. Chapter III deals with the derivation of the split-Levinson algorithms preceded by an introductory section on symmetric polynomials. As in Chapter II, both the autocorrelation function and the lattice algorithms will be developed. In addition, a comparison between the computational cost of the Levinson and split-Levinson algorithms and an attempt to define the split lattice structure in terms similar to the Levinson based lattice are presented.

In the final chapter two practical applications of the split-Levinson algorithm are investigated. These are: (1) the MA parameter estimation problem, and (2) the extended Prony method. In case (1), the Levinson recursion used to determine a predictor coefficient vector is replaced by the split-Levinson algorithm. A comparison between the test coefficients and the computed coefficients is presented. In case (2), an estimation of sinusoids in white noise is performed. Additionally, overall conclusions of the research as well as proposed topics for continued thesis research are presented.

II. THE CLASSICAL LEVINSON ALGORITHMS

The importance of the Levinson algorithm in linear prediction theory is well known. The reason to present the algorithm in its two forms is twofold: (1) to present certain definitions that will be required later in the development of the split-Levinson algorithms, and (2) to detail the computational complexity of the Levinson algorithm for comparison purposes to the split-Levinson versions of the algorithm. In the context of our discussion within this thesis, we shall confine ourselves to the study of autoregressive modeling problems of real sequences as in Figure 1. [Ref. 3: p. 152]

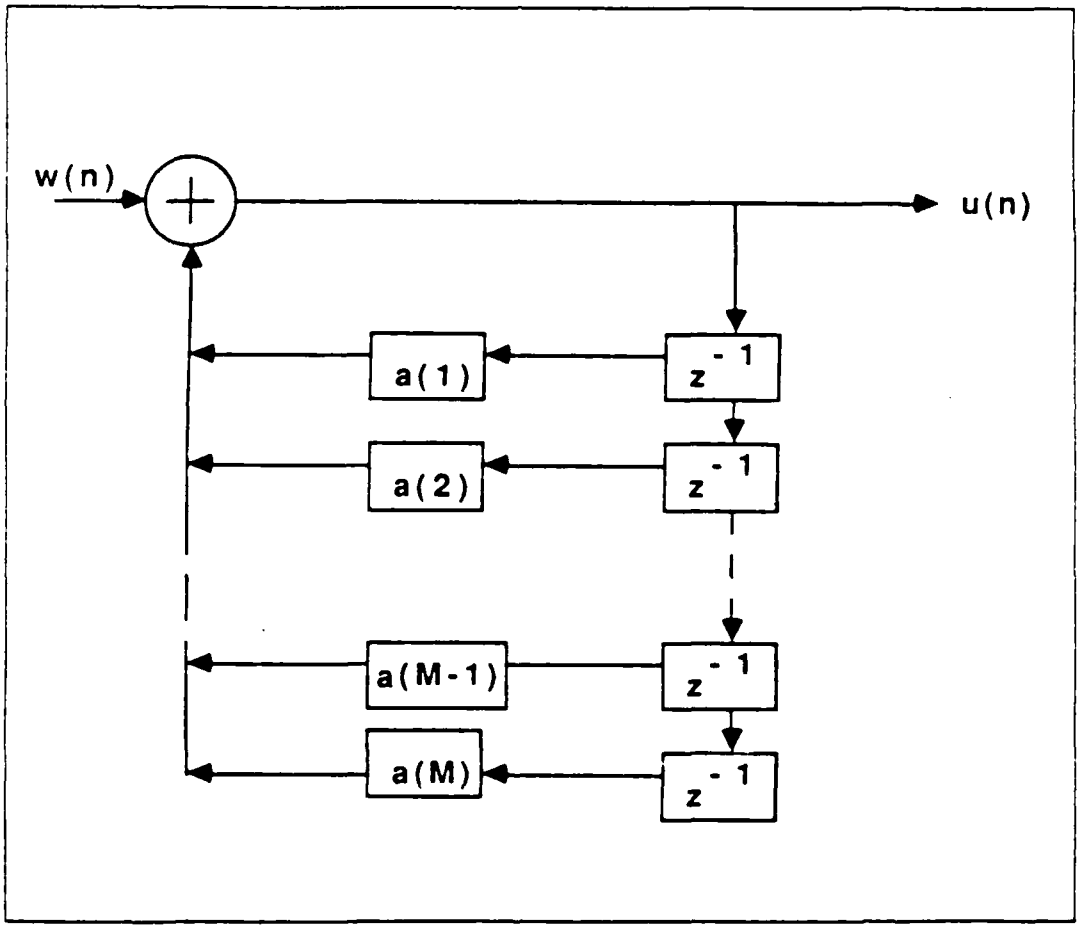


Figure 1. Autoregressive Model.

We know from linear prediction theory the augmented normal equations given by,

$$R_x \mathbf{a} = \mathbf{r}_{xy} \tag{2.1}$$

can optimally be solved by the Levinson algorithm, and that this algorithm can be implemented with either the autocorrelation function or the forward and backward prediction error vectors of the input sequence [Ref. 3: pp. 152-170].

A. THE LEVINSON ALGORITHM

In order to examine the computational complexity of the Levinson recursion, it is necessary to formulate the recursive algorithm, and to determine the number of real multiplications and additions required to complete the algorithm. First, construct a Toeplitz matrix from the sequence s(t), of length N, defined as $R_k = [R_{i,j}; 0 \le i, j \le k]$, where the elements of the matrix are the autocorrelation lags given by [Ref. 2: p. 646]

$$R_i = \frac{1}{N-1-i} \sum_{t=0}^{N-1-i} s(t)\, s(t+i) \tag{2.2}$$

then, the predictor vector **a** can be determined as a solution to the system defined by the matrix equations

$$[R_k][\mathbf{a}_k] = [\sigma_k, 0, 0, \ldots, 0]^T \tag{2.3}$$

where σ_k is the prediction error norm, and is defined as

$$\sigma_k = R_0 + \sum_{i=1}^{k} a_{ki} R_i \tag{2.4}$$

It is recalled from linear prediction theory, that given a positive-definite matrix R_k of order k+1, the kth order ***a*** coefficient vector can be computed recursively from the nested Toeplitz submatrices, and their respective successive predictor vectors, **a**. The well-known Levinson recursion formula is this solution, and has the form

$$a_{kl} = a_{k-1,l} + \rho_k a_{k-1,k-i} \qquad i = 0, 1, 2, \ldots, k \tag{2.5}$$

with the conditions that $a_{k,0} = 1$, and $a_{k-1,k} = 0$. The parameters, $\rho_k = a_{k,k}$, are called reflection coefficients, also PARCOR coefficients, because they represent the partial cor-

relation between the zero-th and the k-th element of the prediction vector with the effect of all the intermediate elements removed.[Ref. 4: p. 53]

To construct the Levinson recursion we must use the prediction error norm relationship

$$\sigma_k = (1 - \rho_k^2)\sigma_{k-1} \tag{2.6}$$

and the identity

$$\sigma_{k-1}\rho_k = -\sum_{i=0}^{k-1} R_{k-i}a_{k-1,i} \tag{2.7}$$

to define the recursion variables. Consider the following definition as it applies to the Levinson recursion [Ref. 1: p. 472].

$$\begin{aligned} \lambda_k &= -\sum_{i=0}^{k-1} R_{k-i}a_{k-1,i} \\ &= \sigma_{k-1}\rho_k \end{aligned} \tag{2.8}$$

and solving for ρ_k from Eq.(2.8), we have

$$\rho_k = \frac{\lambda_k}{\sigma_{k-1}} \tag{2.9}$$

The error norm σ_k can be written in terms of the the normalizing term λ_k by rewriting Eq. (2.6), and making a substitution from Eq. (2.9)

$$\begin{aligned} \sigma_k &= (1 - \rho_k^2)\sigma_{k-1} \\ &= \sigma_{k-1} - \rho_k(\sigma_{k-1}\rho_k) \\ &= \sigma_{k-1} - \rho_k\lambda_k \end{aligned} \tag{2.10}$$

Combining Eqs. (2.5), (2.8), (2.9), and (2.10), we have the basis for the Levinson algorithm, and it is summarized in Table 2 of Appendix A.

B. LEVINSON LATTICE REALIZATION

If we are given a real sequence of signal values s(0) s(1), ..., s(N-1), and it is known that $s(t) = 0$, for $-l \geq t$ and $t \geq N$, then for the linear prediction problem of order **k** we find it necessary to find a set of real numbers $a_{k0}, a_{k1}, \ldots, a_{kk}$ that will minimize the forward and backward prediction errorvectors using a linear combination of the past signal

vectors. If we call the forward prediction error vector $f_k(t)$ and the backward error vector $b_k(t)$, and define them in terms of the a_{ki} coefficients, [Ref. 2: p. 646] we have

$$f_k(t) = \sum_{i=0}^{k} a_{ki}\ s(t-i) \tag{2.11}$$

$$b_k(t) = \sum_{i=0}^{k} a_{k,k-i}\ s(t-i) \tag{2.12}$$

then it turns out that the same real numbers, a_{ki}, will provide the solution to either of the forward or backward prediction problems, (i.e., minimize the squared Euclidean norm of both f_k and b_k).

Let σ_k be defined as the squared norm, that is

$$\sigma_k = \|\mathbf{f}_k\|^2 = \mathbf{f}_k^T\mathbf{f}_k \tag{2.13}$$

From Eqs. (2.11) and (2.12) forming the first three terms of each error vector we have the following,

$$f_k(0) = a_{k0}s(0) + a_{k1}s(-1) + \ldots + a_{kk}s(-k) \tag{2.14}$$

$$f_k(1) = a_{k0}s(1) + a_{k1}s(0) + a_{k2}s(-1) + \ldots + a_{kk}s(1-k) \tag{2.15}$$

$$f_k(2) = a_{k0}s(2) + a_{k1}s(1) + a_{k2}s(0) + \ldots + a_{kk}s(2-k) \tag{2.16}$$

and,

$$b_k(0) = a_{kk}s(0) + a_{k,k-1}s(-1) + a_{k,k-2}s(-2) + \ldots + a_{k0}s(-k) \tag{2.17}$$

$$b_k(1) = a_{kk}s(1) + a_{k,k-1}s(0) + a_{k,k-2}s(-1) + \ldots + a_{k0}s(1-k) \tag{2.18}$$

$$b_k(2) = a_{kk}s(2) + a_{k,k-1}s(1) + a_{k,k-2}s(0) + \ldots + a_{k0}s(2-k) \tag{2.19}$$

If we examine the elements of these two vectors we can see they are related in that each can be derived from the other by reversing the order of the a coefficients. If we form the Euclidean norm of each vector, $\|\mathbf{f}_k\|$ and $\|\mathbf{b}_k\|$, we see that the k-th predictor vector $[a_{k0}, a_{k1},\ldots,a_{kk}]^T$ minimizes the error norm, and $\|f_k\| = \|b_k\|$.

From [Ref. 3: pp. 156-157], we can use the Levinson algorithm to define the recursion formula for the forward and backward prediction errors given by

$$\begin{aligned} f_k(t) &= f_{k-1}(t) + \rho_k b_{k-1}(t-1), \\ b_k(t) &= \rho_k f_{k-1}(t) + b_{k-1}(t-1) \end{aligned} \tag{2.20}$$

If we let the following definition apply to the lattice version of the Levinson algorithm

$$\lambda_k = \sigma_{k-1}\rho_k = -\sum_{t=1}^{N+k-2} f_{k-1}(t) b_{k-1}(t) \tag{2.21}$$

then using Eqs. (2.9), (2.10), (2.20), and (2.21) we can summarize the Levinson lattice algorithm in Table 3 of Appendix A.

Even though the lattice algorithm is implemented directly from the data samples, its computer implementation will be more complex because of the vector manipulations that must occur in each iteration. The lattice structure defined by Eq. (2.20) is shown in Figure 2.

In summary, we discussed the Levinson algorithm which uses the autocorrelation elements in its recursion and the related lattice structure which uses the input data directly in its formulation. In terms of the computational complexity, both algorithms require real multiplications of the order k^2, as detailed in Tables 2 and 3 of Appendix A, in order to realize a k^{th} order predictor filter.

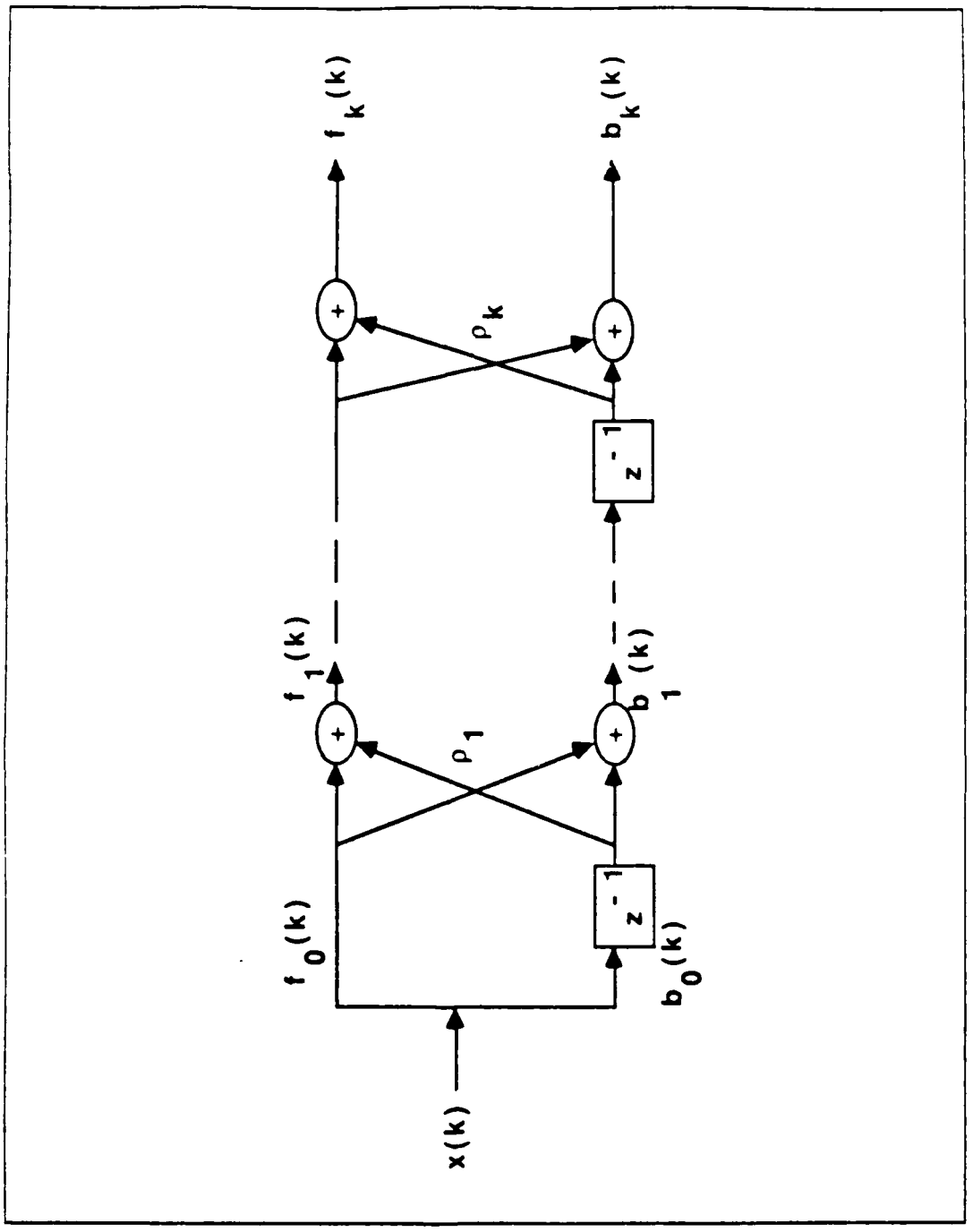


Figure 2. Lattice Prediction Error Filter.

III. THE SPLIT-LEVINSON ALGORITHMS

The split-Levinson algorithms are based on the theory of symmetric and antisymmetric polynomials. We know that for an k-th order real autoregressive process, the normal equations are

$$\begin{bmatrix} R_0 & R_1 & R_2 & \cdots & R_k \\ R_1 & R_0 & R_1 & \cdots & R_{k-1} \\ R_2 & R_1 & R_0 & \cdots & R_{k-2} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ R_k & R_{k-1} & R_{k-2} & \cdots & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ \cdots \\ \cdots \\ \cdots \\ a_k \end{bmatrix} = \begin{bmatrix} \sigma \\ 0 \\ 0 \\ \cdots \\ \cdots \\ \cdots \\ 0 \end{bmatrix} \tag{3.1}$$

or,

$$R\mathbf{a}_k = [\sigma,0,0,\ldots,0]^T \tag{3.2}$$

Using the Barlett Bisection Theorem [Ref. 5: pp. 1074-1076], and because of the symmetry of the autocorrelation matrix, we can say that the predictor coefficient vector is the linear combination of a symmetric and antisymmetric predictor vector given by

$$\mathbf{a}_k = \mathbf{a}_k^{(s)} + \mathbf{a}_k^{(a)} \tag{3.3}$$

The symmetric and antisymmetric vectors are defined as

$$\begin{aligned} \mathbf{a}_k^{(s)} &= \begin{bmatrix} B \\ B' \end{bmatrix} \\ \mathbf{a}_k^{(a)} &= \begin{bmatrix} B \\ -B' \end{bmatrix} \end{aligned} \tag{3.4}$$

where B represents one-half of the vector components of $\mathbf{a}_k$, and B′ represents the reversal of the vector components of B. Using Eqs. (3.3) and (3.4), we can transform the normal equations into

$$R\mathbf{a}_x^{(s)} = [\sigma/2,0,0,\ldots,\sigma/2]^T$$
$$R\mathbf{a}_x^{(a)} = [\sigma/2,0,0,\ldots,-\sigma/2]^T \tag{3.5}$$

Therefore, we can see some favorable consequences of these revised normal equations, and their solutions. First, either the symmetric or antisymmetric form will give the same solution, and second, because of the symmetry of the predictor vectors, we need only solve for one-half of the predictor coefficients.

Similar to the Levinson algorithm we now proceed to develop the split-Levinson algorithms from the input sequence autocorrelation function and the predictor error vectors.

A. SPLIT-LEVINSON ALGORITHM

The predictor polynomial $a_k(z)$ is defined as

$$a_k(z) = \sum_{i=0}^{k} a_{ki} z^{-i} \tag{3.6}$$

relative to the given Toeplitz matrix of autocorrelation lags. Denote the reverse of our predictor polynomial as $\hat{a}(z) = z^{-k}a_k(z^{-1})$, and the predictor polynomial has been shown to obey the recursion [Ref. 3: pp. 156-157]

$$a_k(z) = a_{k-1}(z) + \rho_k z^{-1}\hat{a}(z) \tag{3.7}$$

and the reverse polynomial of Eq. (3.7) is

$$\hat{a}_k(z) = z^{-1}a_{k-1}(z) + \rho_k a_{k-1}(z) \tag{3.8}$$

We now want to form a new polynomial from the given predictor polynomial that will form the basis of the split-Levinson algorithm. It is desired to show that the determination of the coefficients of this polynomial will allow us to recover the original predictor polynomial, and at the same time be more computationally efficient. We define the symmetric polynomial as $P_k(z)$, and the antisymmetric polynomial as $P_k^{(a)}(z)$, and we desire them to be of the form [Ref. 1: p. 472]

$$P_k(z) = \sum_{i=0}^{k} p_{ki} z^{-i}$$
$$P_k^{(a)}(z) = \sum_{i=0}^{k} p_{ki}^{(a)} z^{-i} \tag{3.9}$$

Recall from Eq. (3.4) and (3.5) that the symmetric and antisymmetric predictor coefficients are composed of two vectors that are reverses of each other, and we will define these vectors so that they obey the relationships

$$p_{ki} = p_{k,k-i}$$
$$p_{ki}^{(a)} = -p_{k,k-i}^{(a)} \tag{3.10}$$

Consider the mathematical interpretation of making the autocorrelation matrix, R_k a singular matrix. If the reflection coefficient ρ_k is made ± 1, then this corresponds to an element of R_k making the matrix singular. For this reason we shall designate the symmetric and antisymmetric predictor polynomials as *singular* predictor polynomials [Ref. 2: p. 472] and from Eq. (3.9) they are defined as

$$P_k(z) = a_{k-1}(z) + z^{-1}\hat{a}_{k-1}(z)$$
$$P_k^{(a)}(z) = a_{k-1}(z) - z^{-1}\hat{a}_{k-1}(z) \tag{3.11}$$

Also, these singular predictor polynomials are self-reciprocal [Ref. 2: p. 472] because of their symmetry and may be expressed in the following forms

$$P_k(z) = z^{-k} P_k(z^{-1})$$
$$P_k^{(a)} = -z^{-k} P_k(z^{-1}) \tag{3.12}$$

From Eq. (3.11) we have

$$\begin{aligned} z^{-1}\hat{a}_{k-1}(z) &= P_k(z) - a_{k-1}(z) \\ &= a_{k-1}(z) - P_k^{(a)}(z) \end{aligned} \tag{3.13}$$

If we add Eqs. (3.7) and (3.8), and make a substitution from Eq. (3.13) we have

$$\begin{aligned} a_k(z) + \hat{a}_k(z) &= z^{-1}\hat{a}_{k-1}(z) + \rho_k a_{k-1}(z) + a_{k-1}(z) + \rho_k \hat{a}_{k-1}(z) \\ &= (1+\rho_k)a_{k-1}(z) + (1+\rho_k)z^{-1}\hat{a}_{k-1}(z) \\ &= \lambda_k P_k(z) \end{aligned} \tag{3.14}$$

where we have defined λ_k as

$$\lambda_k = 1 + \rho_k \tag{3.15}$$

In a similar fashion we can solve for the antisymmetric normalized singular predictor polynomial by subtracting Eqs. (3.7) and (3.8), and substituting from Eq. (3.13) we have

$$a_k(z) - \hat{a}_k(z) = \lambda_k^{(a)} P_k^{(a)}(z) \tag{3.16}$$

where

$$\lambda_k^{(a)} = 1 - \rho_k \tag{3.17}$$

Similar to the predictor polynomial $a_k(z)$, we can define the singular predictor coefficient vectors for Eq. (3.9) as [Ref. 2 : p. 472]

$$\begin{aligned} \mathbf{p}_k &= [p_{k0}, p_{k1}, \cdots, p_{kk}]^T \\ \mathbf{p}_k^{(a)} &= [p_{k0}^{(a)}, p_{k1}^{(a)}, \cdots, p_{kk}^{(a)}]^T \end{aligned} \tag{3.18}$$

Since we want the split-Levinson normal equations to be of the form

$$R_k \mathbf{a}_k = [\sigma_k, 0, 0, \ldots, 0]^T \tag{3.19}$$

$$R_k \hat{\mathbf{a}}_k = [0, 0, \ldots, \sigma_k]^T \tag{3.20}$$

then from Eq. (3.14) and (3.16) the singular predictor polynomials are

$$\begin{aligned} P_k(z) &= \frac{a_k(z)}{\lambda_k} + \frac{\hat{a}_k(z)}{\lambda_k} \\ P_k^{(a)}(z) &= \frac{a_k(z)}{\lambda_k^{(a)}} - \frac{\hat{a}_k(z)}{\lambda_k^{(a)}} \end{aligned} \tag{3.21}$$

Since $a_k(z)$ is a polynomial formed from the predictor coefficient vector that is a solution to Eq. (2.3), it follows that $P_k(z)$ and $P_k^{(a)}(z)$ are solutions to the Toeplitz system described by Eqs. (3.19) and (3.20). [Ref. 1: p. 472]

$$R_k \mathbf{p}_k = R_k \left[\frac{\mathbf{a}_k}{\lambda_k} + \frac{\hat{\mathbf{a}}_k}{\lambda_k} \right]$$
$$R_k \mathbf{p}_k^{(a)} = R_k \left[\frac{\mathbf{a}_k}{\lambda_k^{(a)}} - \frac{\hat{\mathbf{a}}_k}{\lambda_k^{(a)}} \right] \tag{3.22}$$

Normalizing Eqs. (3.19) and (3.20) by λ_k and $\lambda_k^{(a)}$, the split-Levinson normal equations in matrix form are expressed as

$$R_k \mathbf{p}_k = [\tau_k, 0, 0, \ldots, \tau_k]^T$$
$$R_k \mathbf{p}_k^{(a)} = [\tau_k^{(a)}, 0, 0, \ldots, -\tau_k^{(a)}]^T \tag{3.23}$$

where we have defined the modified prediction error norm [Ref. 2: p. 651]

$$\tau_k = \frac{\sigma_k}{\lambda_k}$$
$$\tau_k^{(a)} = \frac{\sigma_k}{\lambda_k^{(a)}} \tag{3.24}$$

If we expand the matrix expressions in Eq. (3.23), the modified error norms may be expressed as finite sums of the predictor coefficients

$$\tau_k = \sum_{i=0}^{k} R_i p_{ki}$$
$$\tau_k^{(a)} = \sum_{i=0}^{k} R_i p_{ki}^{(a)} \tag{3.25}$$

where R_i is the i-th autocorrelation element of the k-th sub matrix.

Since the symmetric and antisymmetric polynomials are closely related, we shall derive only the symmetric polynomial recursion equations, and then simply present the results for the antisymmetric case.

The final step in the derivation is to derive a three term recursion formula for the symmetric polynomial. From Eq. (3.11) and (3.14) we have the surprising result that the predictor polynomial $a_k(z)$ can be obtained from a linear combination of successive singular predictor polynomials [Ref. 2: p. 472]. First, form $P_{k+1}(z)$ from Eq. (3.11), and then eliminate $\hat{a}_k(z)$ using Eq. (3.14)

$$
\begin{aligned}
P_{k+1}(z) &= a_k(z) + z^{-1}\hat{a}_k(z) \\
&= a_k(z) + z^{-1}[\lambda_k P_k(z) - a_k(z)] \\
&= (1 - z^{-1})a_k(z) + z^{-1}\lambda_k P_k(z) \\
(1 - z^{-1})a_k(z) &= P_{k+1}(z) - z^{-1}\lambda_k P_k(z)
\end{aligned} \tag{3.26}
$$

If we replace k by k-1 in Eq. (3.26) above we also have

$$
(1 - z^{-1})a_{k-1}(z) = P_k(z) - z^{-1}\lambda_{k-1}P_{k-1}(z) \tag{3.27}
$$

We now form our recursion formula by mutiplying Eq. (3.11) by $(1 - z^{-1})$, and use Eqs. (3.15), (3.26), and (3.27) to eliminate ρ_k and all a_k predictor polynomials.

$$
\begin{aligned}
(1 - z^{-1})a_k(z) &= (1 - z^{-1})a_{k-1}(z) + \rho_k(1 - z^{-1})z^{-1}\hat{a}_{k-1}(z) \\
&= (1 - z^{-1})a_{k-1}(z) + \rho_k(1 - z^{-1})[P_k(z) - a_{k-1}(z)] \\
&= (1 - z^{-1})a_{k-1}(z) + \rho_k[P_k(z) - a_{k-1}(z) - z^{-1}P_k(z) + z^{-1}a_{k-1}(z)] \\
&= a_{k-1}(z)[1 - z^{-1} - \rho_k + z^{-1}\rho_k] \\
&= a_{k-1}(z)[(1 - z^{-1})(1 - \rho_k)]
\end{aligned} \tag{3.28}
$$

If we now substitute for $(1 - z^{-1})a_k(z^{-1})$, $(1 - z^{-1})a_{k-1}(z^{-1})$ from Eqs. (3.27) and (3.28), and eliminate ρ_k using Eq. (3.15), we can complete the derivation

$$
\begin{aligned}
P_{k+1}(z) - z^{-1}\lambda_k P_k(z) &= [P_k(z) - z^{-1}\lambda_{k-1}P_{k-1}(z)](2 - \lambda_k) + [\lambda_k P_k(z) - P_k(z)](1 - z^{-1}) \\
&= 2P_k(z) - \lambda_k(z)P_k(z) - 2z^{-1}\lambda_{k-1}P_{k-1}(z) + z^{-1}\lambda_k\lambda_{k-1}P_{k-1}(z) \\
&\quad + \lambda_k P_k(z) - P_k(z) - z^{-1}\lambda_k P_k(z) + z^{-1}P_k(z) \\
P_{k+1}(z) &= (1 - z^{-1})P_k(z) + z^{-1}\lambda_{k-1}P_{k-1}(z)[\lambda_k - 2] \\
&= (1 + z^{-1})P_k(z) - \alpha_k z^{-1}P_{k-1}(z)
\end{aligned} \tag{3.29}
$$

Taking the inverse Z transform of Eq. (3.29), we have the three term recursion formula for the singular predictor coefficients

$$
p_{ki} = p_{kl} + p_{k,k-i} - \alpha_k p_{k-1,k-i} \tag{3.30}
$$

where the recursion parameter α_k is defined as

$$
\alpha_k = \lambda_{k-1}[2 - \lambda_k] \tag{3.31}
$$

We note that τ_k is determined from $P_k(z)$ from Eq. (3.23), and therefore we conclude that the singular predictor polynomials can be recursively computed from Eq. (3.30). How-

ever, the recursion parameter α_k is not quite in the correct form. From Eqs. (2.10), (3.15), and (3.31) we can alternatively compute α_k from [Ref. 2: p. 473]

$$\alpha_k = \frac{\tau_k}{\tau_{k-1}} \tag{3.32}$$

The dual relationships for the antisymmetric split-Levinson formulas can be derived by following a procedure similar to the one presented above. It suffices to replace the quantities $\lambda_k, \tau_k, \alpha_k, p_{k,i}$ by their antisymmetric duals, i.e., $p_{k,i}^{(a)}$, and use the following antisymmetric initial conditions. [Ref. 2: p. 649]

$$\begin{aligned} p_{00}^{(a)} &= 0 \\ p_{10}^{(a)} &= 1 \\ p_{11}^{(a)} &= -1 \\ \tau_0^{(a)} &= R_0 \end{aligned} \tag{3.33}$$

Recursive equations for the symmetric split-Levinson algorithm are summarized in Table 4 of Appendix A. Examining the entries in Table 4, we see that a full iteration loop of the algorithm requires approximately t real multiplications. However, because of the symmetry of the singular predictor coefficients, we only have to perform one-half of these calculations. Therefore, for a k-th order filter we need to make on the order of $k^2/2$ real multiplications. The δ function in Table 4 is used to distinguish between even and odd orders of the indexing variable.

The FORTRAN program SPLIT, in Appendix B, estimates the predictor coefficients using the Levinson and split-Levinson algorithms. Figure 3 is a graphical comparison between the known test filter coefficients of SPLIT, shown by the solid curve, and the filter coefficients computed by the Levinson and split-Levinson algorithms, shown by the dashed curve. We now undertake the derivation of the lattice form of the split-Levinson formulas to verify the numerical complexity of that method, and to investigate the symmetric and antisymmetric lattice structures compared to the Levinson lattice forms.

B. SPLIT LATTICE ALGORITHM

We begin the split lattice derivation by introducing the symmetric and antisymmetric error vectors $\mathbf{x}_k$, $\mathbf{x}_k^{(a)}$ [Ref. 2: p. 648]. If we use previously established singularity concepts, and substitute ± 1 for ρ_k in Eq. (2.20) for the symmetric and antisymmetric error vectors, $\mathbf{x}_k$ and $\mathbf{x}_k^{(a)}$, respectively, we have

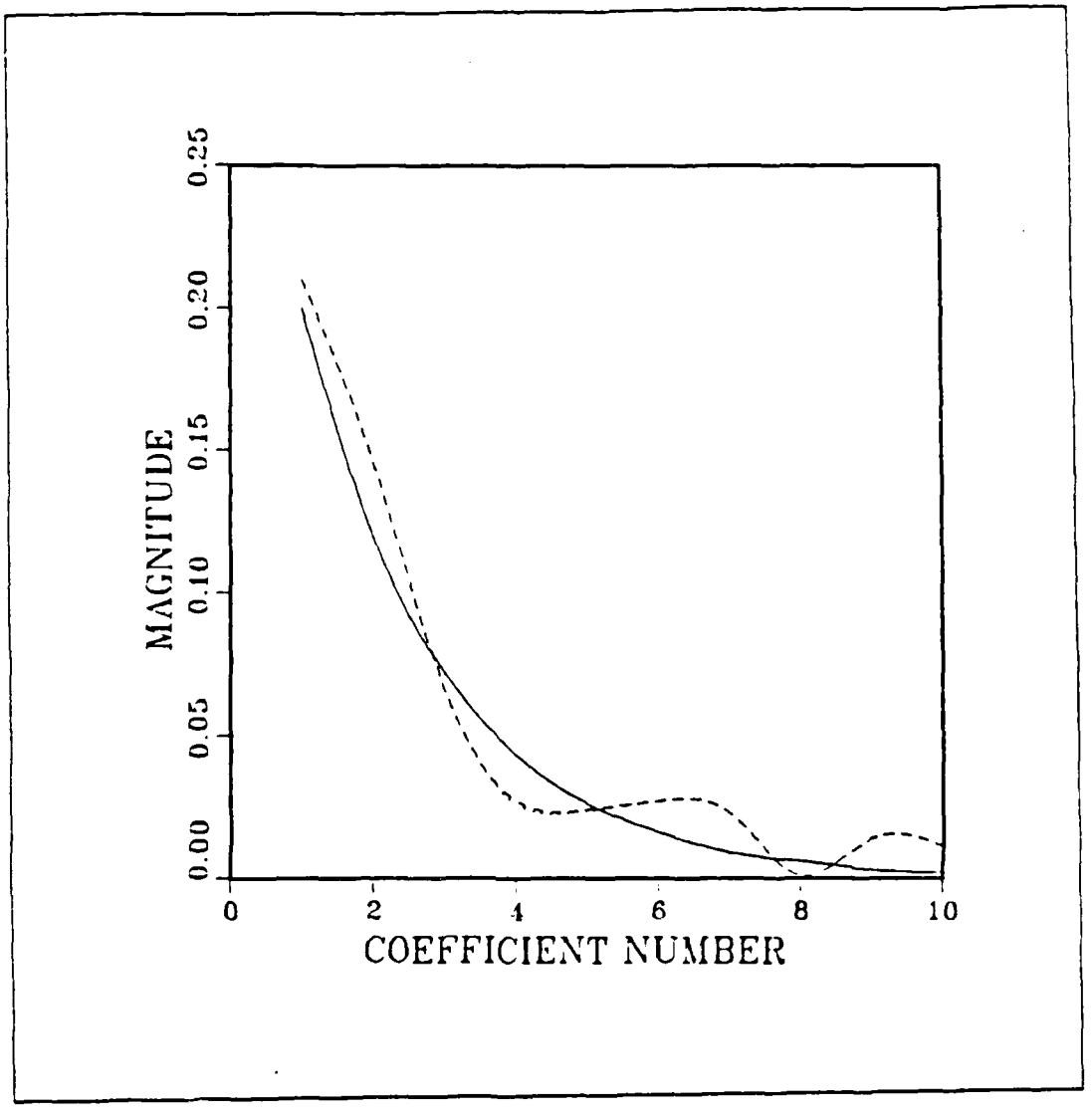


Figure 3. Levinson / Split Levinson Coefficient Comparison.

$$x_k(t) = f_{k-1}(t) + b_{k-1}(t-1)$$
$$x_k^{(a)}(t) = f_{k-1}(t) - b_{k-1}(t-1) \tag{3.34}$$

As in the split-Levinson case, we shall proceed with the derivation of the symmetric split lattice, and present the antisymmetric lattice results at the end of the derivation with any significant changes noted.

If we extend the singular polynomial concept to the singular predictor coefficients, we can start with the Levinson coefficient recursion formula and substitute ± 1 for ρ_k

$$a_{ki} = a_{k-1,i} + \rho_k a_{k-1,k-i} \tag{3.35}$$

and substituting for ρ_k

$$p_{ki} = a_{k-1,i} + a_{k-1,k-i} \tag{3.36}$$

If we write Eq. (3.34) representing the time index (t) with the subscript (i) we have an algorithm that is more easily adapted for computers.

$$x_{ki} = f_{k-1,i} + b_{k-1,k-i} \tag{3.37}$$

Now, comparing Eqs. (3.36) and (3.37) we have a direct correlation between the two, and from the split-Levinson equation for the forward error vector, we can write the dual split lattice equation for $x_k(t)$

$$x_k(t) = \sum_{i=0}^{k} p_{ki}\, s(t-i) \tag{3.38}$$

Since Eq. (3.38) is in the form of a convolution sum we can apply Z transform theory to see if any inferences can be made

$$\begin{aligned} X_k(z) &= P_k(z)S(z) \\ \frac{X_k(z)}{S(z)} &= P_k(z) \end{aligned} \tag{3.39}$$

From Eq. (3.39) we can conclude that the symmetric polynomial constitutes the Z transform of the transfer function formed from the error vector and input sequence z polynomials. Now we can use the previously derived split-Levinson algorithm, and inverse transform it to obtain the lattice error vector recursion algorithm.

Repeating the split-Levinson recursion we have

$$\begin{aligned} P_{k+1}(z) &= (1 + z^{-1})P_k(z) - \alpha_k z^{-1} P_{k-1}(z) \\ &= P_k(z) + z^{-1} P_k(z) - \alpha_k z^{-1} P_{k-1}(z) \end{aligned} \tag{3.40}$$

Multiplying Eq. (3.40) by S(z) and using Eq. (3.39) for $P_k(z)S(z)$ we have

$$\begin{aligned} S(z)P_{k+1}(z) &= S(z)[P_k(z) + z^{-1}P_k(z) - \alpha_k P_{k-1}(z)] \\ X_{k+1}(z) &= X_k(z) + z^{-1}X_k(z) - \alpha_k z^{-1}X_{k-1}(z) \end{aligned} \tag{3.41}$$

Applying the inverse Z transform to each side of Eq. (3.41) we have the singular predictor error vector recursion formula [Ref. 2: p. 650]

$$x_{k+1}(t) = x_k(t) + z^{-1}x_k(t) - \alpha_k z^{-1}x_{k-1}(t) \tag{3.42}$$

The symmetric lattice structure given by Eq. (3.42) is shown by Figure 4. [Ref. 2: p. 650]

From Eq. (3.32) we know that the recursion parameter α_k is defined as τ_k/τ_{k-1}, and since α_k appears in the recursion formula for the singular error vector, we need to solve for it. We begin with the Levinson error norm equation

$$\begin{aligned} \sigma_k &= (1 - \rho_k^2)\sigma_{k-1} \\ 2\sigma_k &= 2\sigma_{k-1}(1 + \rho_k)(1 - \rho_k) \\ 2\frac{\sigma_k}{1 + \rho_k} &= 2\sigma_{k-1}(1 - \rho_k) \\ 2\sigma_{k-1}(1 - \rho_k) &= 2\tau_k \end{aligned} \tag{3.43}$$

where we have substituted τ_k from Eq. (3.32), and $2\sigma_{k-1}(1 - \rho_k)$ is defined as $\|x_k\|^2$ [Ref. 2: p. 650].

The initial conditions for Eq. (3.42) must be examined because most cases are trivial except for the case of k = 0. From Eq. (3.38) we have

$$x_0(t) = p_{00}s(t), \tag{3.44}$$

and from [Ref. 2: p. 648] we define $p_{00} = 2$.

The antisymmetric duals are very similar to the symmetric case, and can be formed by replacing the symmetric variable by its antisymmetric counterpart, i.e., $x_k^{(a)}(t)$ for $x_k(t)$, $-\rho_k$ for ρ_k, etc. The initial conditions for the antisymmetric case are given below

$$\begin{aligned} x_0^{(a)}(t) &= 0 \\ x_k^{(a)}(t) &= s(t) - s(t-1) \end{aligned} \tag{3.45}$$

The antisymmetric lattice structure given by the antisymmetric dual of Eq. (3.42) is shown in Figure 5. The split-Levinson lattice formulas given by Eqs. (3.37), (3.43), and (3.44) are summarized in Table 5 of Appendix A. If we examine the number of real

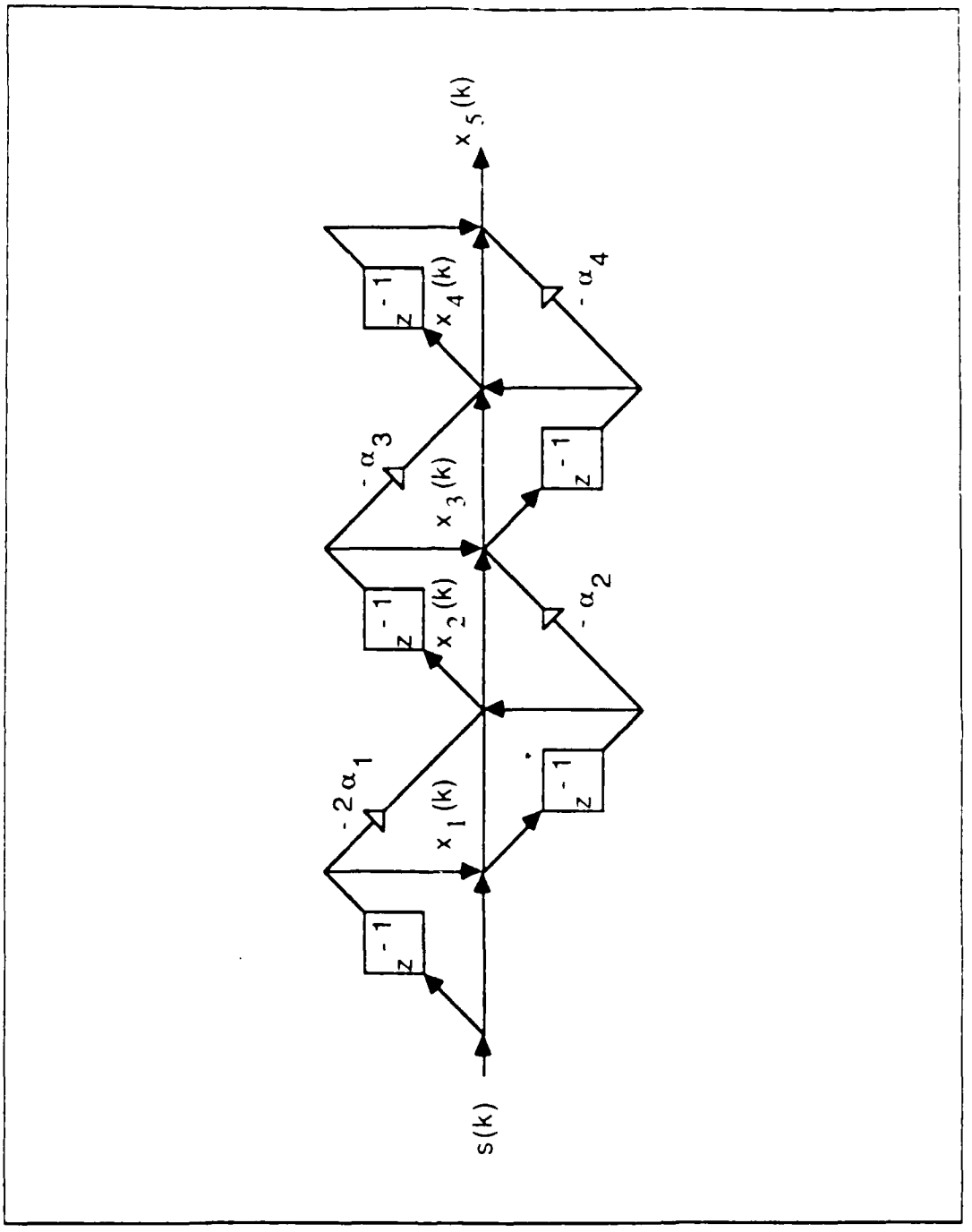


Figure 4. Symmetric Lattice Structure.

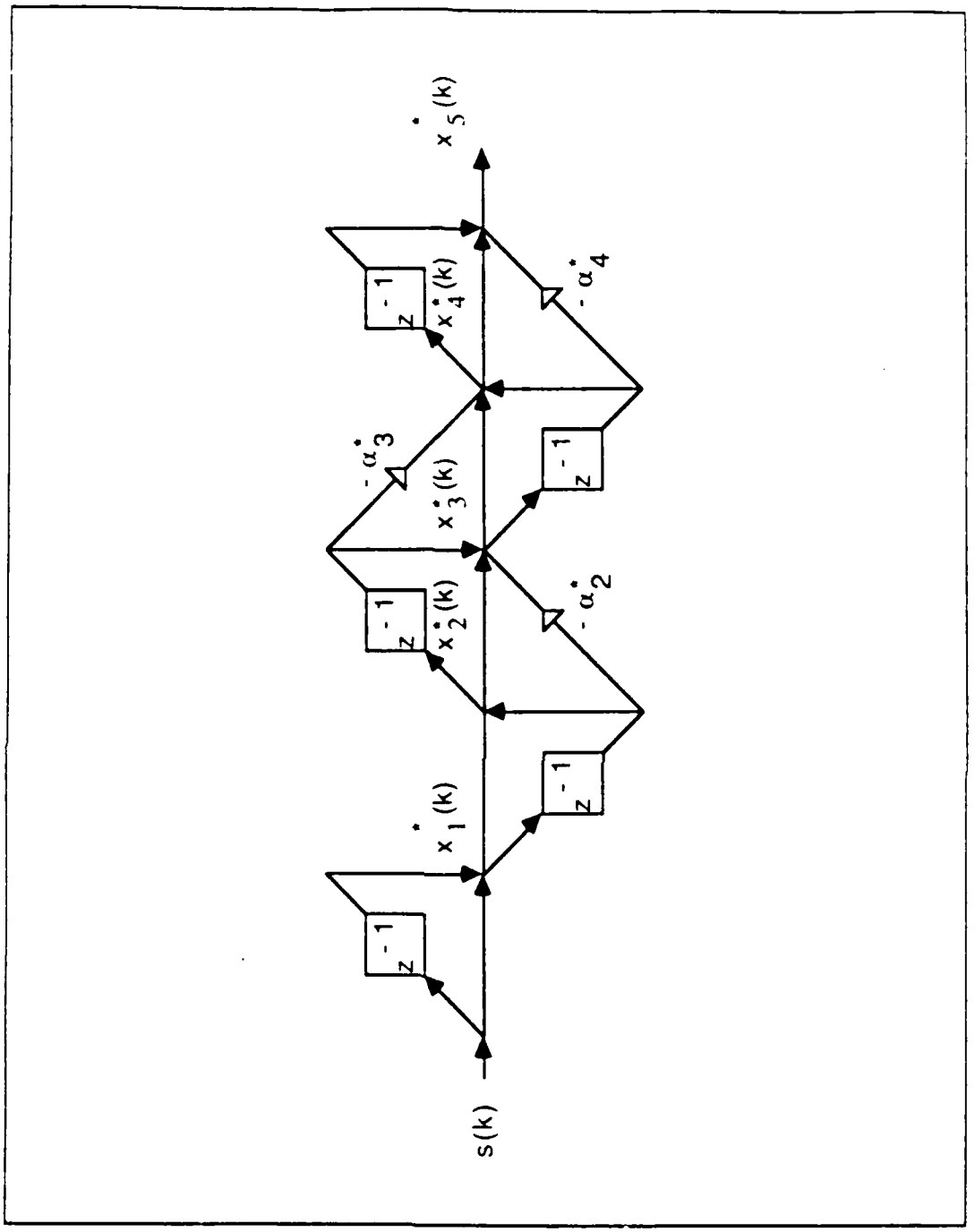


Figure 5. Antisymmetric Lattice Structure.

multiplications given in Table 4 and Table 5, then we can deduce the following conclusions. The split versions of the Levinson algorithm do produce a reduction in the complexity of the calculations when compared to the classical versions. The split-Levinson produces a reduction of one-half, and the Levinson produces a reduction of 3/2 [Ref. 2: p. 645]. The FORTRAN program SLATIS, Appendix C, implements the Levinson and split-Levinson lattice algorithms, and a graphical comparison between the known test coefficients, shown by the solid curve, the coefficients estimated by the Levinson lattice algorithm, shown by the dashed curve, and the coefficients estimated by the symmetric split-Levinson algorithm, shown by the dotted curve, is presented in Figure 6.

The split lattice structures shown in Figure 4 on page 19, and in Figure 5 on page 20 show that the classical lattice structure appears to be lost in the new algorithm. The distinct advantage of the original lattice structure is the modularity of the filter. In order to retrieve this appealing feature of the lattice filter we shall now proceed to derive a revised version of the split lattice structure, and see if it can have a form similar to the classical structure.

C. SPLIT LATTICE REVISED STRUCTURE

To begin, consider the second order classical lattice structure derived from Figure 2 on page 8. We can write the following equations for the first and second stage forward and backward prediction errors,

$$\begin{aligned} f_1(n) &= s(n) + \rho_1 s(n-1) \\ g_1(n) &= \rho_1 s(n) + s(n-1) \\ f_2(n) &= f_1(n) + \rho_2 g_1(n-1) \\ g_2(n) &= \rho_2 f_1(n) + g_1(n-1) \end{aligned} \tag{3.46}$$

Now substituting the equations for $f_1(n)$ and for $g_1(n)$ into the equations for $f_2(n)$ and $g_2(n)$, solving for the second stage forward and backward prediction filter errors and taking the Z transforms yields the transfer functions

$$\frac{F_2(z)}{S(z)} = 1 + z^{-1}(\rho_1 + \rho_1\rho_2) + \rho_2 z^{-2} \tag{3.47}$$

and,

$$\frac{G_2(z)}{S(z)} = \rho_2 + z^{-1}(\rho_1 + \rho_1\rho_2) + z^{-2} \tag{3.48}$$

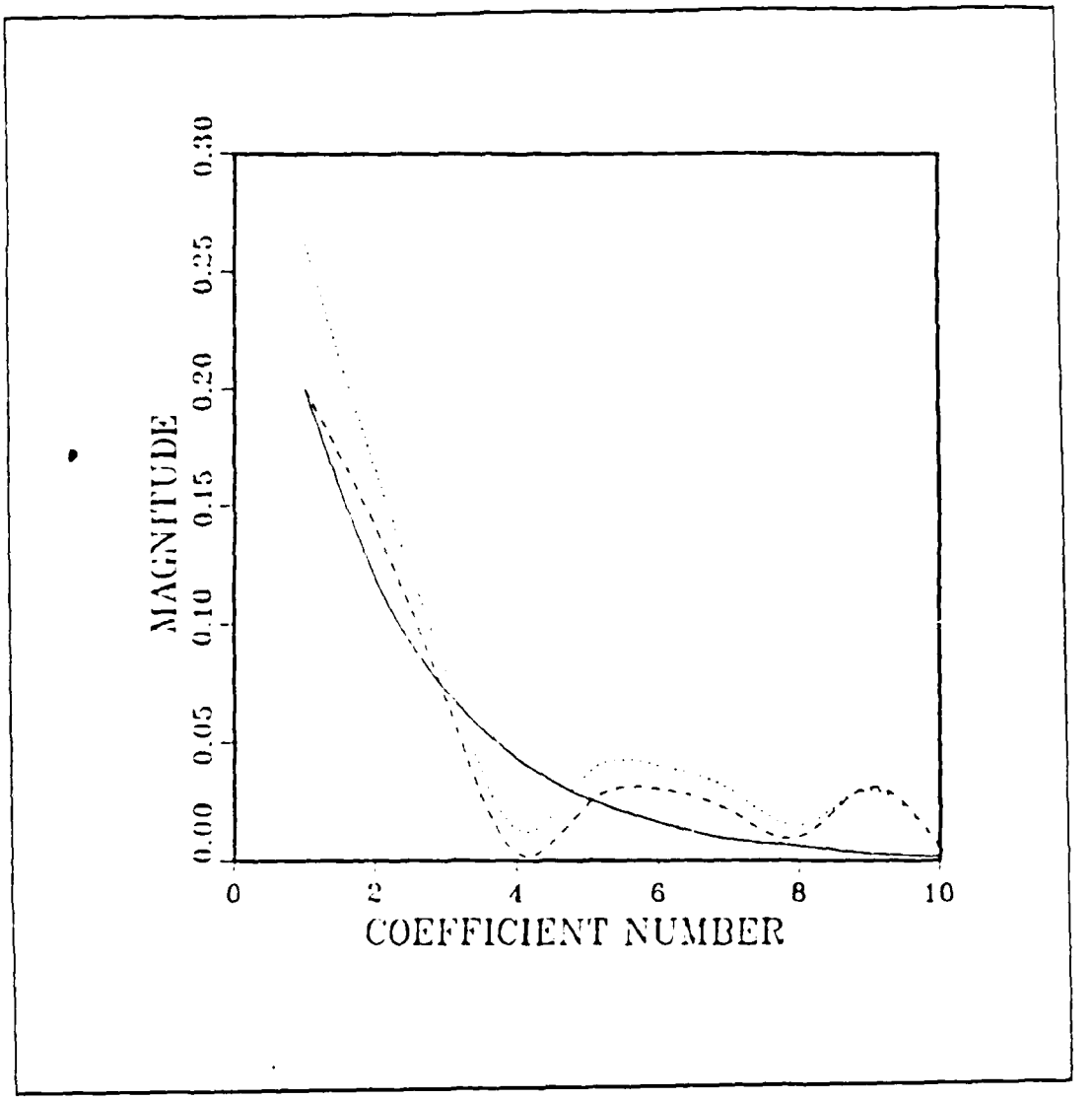


Figure 6. Levinson vs. split-Levinson Coefficient Comparison.

which obviously yield the second order predictor transfer function $A_2(z)$ and its reverse version $\dot{A}_2(z)$, where $a_{20} = 1$, $a_{22} = \rho_2$, $a_{22} = \rho_1 - \rho_1\rho_2 = \rho_1(1 - \rho_2)$. Now forming the third order symmetric polynomial $P_3(z)$ from our second order example, we have [Ref. 1: p. 472]

$$\begin{aligned} P_3(z) &= A_2(z) + z^{-3}A_2(z^{-1}) \\ &= a_0 + (a_1 + a_2)z^{-1} + (a_1 + a_2)z^{-2} + a_0z^{-3} \end{aligned} \tag{3.49}$$

If we define $a_0 = 1$, and $(a_1 + a_2) = p_1$, then

$$\begin{aligned} P_3(z) &= 1 + p_1 z^{-1} + p_1 z^{-2} + z^{-3} \\ &= 1 + p_1 z^{-1} + z^{-2}(p_1 + z^{-1}) \end{aligned} \tag{3.50}$$

Define the following terms:

$$\begin{aligned} F_1(z) &= 1 + p_1 z^{-1} \\ G_1(z) &= z^{-1}(1 + p_1 z) \end{aligned} \tag{3.51}$$

therefore,

$$P_3(z) = F_1(z) + z^{-2} G_1(z) \tag{3.52}$$

Equation (3.52) defines the revised symmetric split-lattice structure, and Figure 7 gives a graphical representation of that structure.

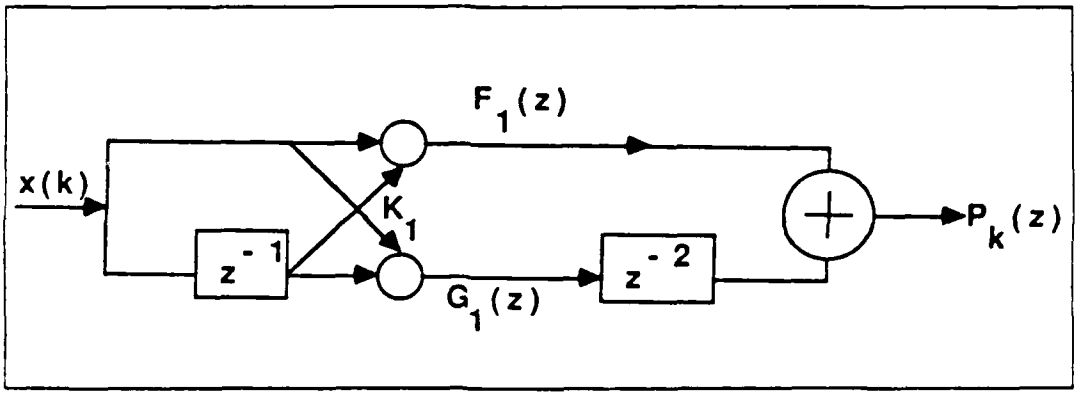


Figure 7. Proposed Split Lattice Configuration

In order to show that the preceding classical structure can be equated with the symmetric polynomial derived earlier, it is now necessary to form the forward and backward prediction error transfer function of the new split lattice structure and compare them to the Z transform of the symmetric polynomial. Therefore, from Figure 7 and Eq. (3.52), we can write $P_3(z)$ as

$$P_3(z) = 1 + K_1 z^{-1} + z^{-2}(K_1 + z^{-1}) \tag{3.53}$$

Now, if we compare Eq. (3.50) to Eq.(3.53), we see that $K_1 = p_1$. But, from Eq. (3.49) we know that $p_1 = (a_1 + a_2)$, and substituting for a_1 and a_2 from Eq. (3.48) and (3.49) yields

$$K_1 = p_1 = \rho_1 + \rho_2(1 - \rho_1) \tag{3.54}$$

Let us now rederive the symmetric polynomial from the revised split lattice structure. From Eq. (3.50) and (3.54) we have

$$P_3(z) = 1 + (\rho_1 - \rho_2 - \rho_1\rho_2)z^{-1} + z^{-2}[(\rho_1 - \rho_2 - \rho_1\rho_2) + z^{-1}] \tag{3.55}$$

Since we have found that the symmetric lattice can be restructured to a form similar to the classical lattice, the next logical step is to find a recurrence relation for the new lattice. Let us consider a 5^{th} order symmetric polynomial to determine the step-down recursion procedure, given by

$$P_5(z) = 1 + p_{51}z^{-1} + p_{52}z^{-2} + p_{52}z^{-3} + p_{51}z^{-4} + z^{-5} \tag{3.56}$$

From Eq. (3.51) we have

$$\begin{aligned} F_2(z) &= 1 + p_{51}z^{-1} + p_{52}z^{-2} \\ G_2(z) &= z^{-2} + p_{51}z^{-1} + p_{52} \end{aligned} \tag{3.57}$$

As per the observations made in Figure 7 on page 23 and Eq. (3.51), we have the lattice reflection coefficient for the second stage, $K_2 = p_{52}$. Now reduce the order of $F(z)$ to find the first stage reflection coefficient using the standard inverse Levinson recursion [Ref. 4: pp. 156-157].

$$\begin{aligned} F_1(z) &= \frac{F_2(z) - K_2G_2(z)}{1 - K_2^2} \\ &= 1 + \frac{p_{51}}{1 + K_2}z^{-1} \end{aligned} \tag{3.58}$$

therefore

$$K_1 = \frac{p_{51}}{1 + K_2} \tag{3.59}$$

Now rewriting the equations for $F_1(z)$ and $F_2(z)$ using the derived reflection coefficients, we have

$$
\begin{aligned}
F_1(z) &= 1 + K_1 z^{-1} \\
G_1(z) &= z^{-1} + K_1 \\
F_2(z) &= 1 + K_1(1 + K_2)z^{-1} + K_2 z^{-2} \\
G_2(z) &= z^{-2} + K_1(1 + K_2)z^{-1} + K_2
\end{aligned}
\tag{3.60}
$$

and we know that

$$
\begin{aligned}
P_5(z) &= F_2(z) + z^{-3} G_2(z) \\
&= 1 + K_1(1 + K_2)z^{-1} + K_2 z^{-2} + K_2 z^{-3} \\
&\quad + K_1(1 + K_2)z^{-1} + z^{-5}
\end{aligned}
\tag{3.61}
$$

Equating the terms in Eq. (3.56) and (3.61), we have $p_{51} = K_1(1 + K_2)$ and $p_{52} = K_2$. Knowing the values of K_1 and K_2, we now can form a two stage symmetric lattice similar to that in Figure 7 on page 23 to implement $P_5(z)$. However, we are interested to find if we can recursively obtain the lower orders $P_4(z)$, $P_3(z)$, etc. or the higher orders $P_6(z)$, $P_7(z)$, etc. from $P_5(z)$. In an attempt to form $P_6(z)$, we use the standard forward recursion [Ref. 3: pp. 156-157] to obtain

$$
\begin{aligned}
F_3(z) &= F_2(z) + K_3 z^{-1} G_2(z) \\
&= 1 + K_1(1 + K_2)z^{-1} + K_2 z^{-2} + K_2 K_3 z^{-1} \\
&\quad + K_1 K_3(1 + K_2)z^{-2} + K_3 z^{-3}
\end{aligned}
\tag{3.62}
$$

and

$$
\begin{aligned}
G_3(z) &= z^{-3} + K_1(1 + K_2)z^{-2} \\
&\quad + K_2 z^{-1} + K_2 K_3 z^{-2} \\
&\quad + K_1 K_3(1 + K_2)z^{-1} + K_3
\end{aligned}
\tag{3.63}
$$

Forming the symmetric polynomial $P_6(z)$ from Eqs. (3.62) and (3.63) we have

$$
\begin{aligned}
P_6(z) &= F_3(z) + z^{-3} G_3(z) \\
&= 1 + (K_1 + K_1 K_2 + K_2 K_3)z^{-1} \\
&\quad + (K_2 + K_1 K_3 + K_1 K_2 K_3)z^{-2} + K_3 z^{-3} \\
&\quad + K_3 z^{-3} + (K_2 + K_1 K_3 + K_1 K_2 K_3)z^{-4} \\
&\quad + (K_1 + K_1 K_2 + K_2 K_3)z^{-5} + z^{-6}
\end{aligned}
\tag{3.64}
$$

Comparing Eq. (3.64) to the symmetric form of $P_6(z)$, we have

$$\begin{aligned} \rho_{61} &= (K_1 + K_1K_2 + K_2K_3) \\ &= \frac{\rho_{51}}{1+\rho_{52}} + \frac{\rho_{51}\rho_{52}}{1+\rho_{52}} + \frac{1}{2}\rho_{52}\rho_{63} \end{aligned} \tag{3.65}$$

The one-half enters into Eq. (3.65) because we know that for even orders the polynomial coefficients are symmetric about the center element, and they must be shared in the matrix equations. We shall now expand Eq. (3.65) to attempt to develop a recursive algorithm for the sixth order coefficients from the fifth order coefficients. Expanding Eq. (3.65) we have

$$(1+\rho_{52})\rho_{61} = \rho_{51} + \rho_{51}\rho_{52} + \left(\frac{1+\rho_{52}}{2}\right)[\rho_{52} + \rho_{52} - \alpha_5\rho_{42}] \tag{3.66}$$

where the term in brackets is an expansion of the coefficient recursion formula, Eq. (3.30), for ρ_{63}. Collecting terms we have

$$\begin{aligned} (1+\rho_{52})\rho_{61} &= \rho_{51}(1+\rho_{52}) - (1+\rho_{52})\frac{1}{2}\rho_{52}[\rho_{52} + \rho_{52} - \alpha_5\rho_{42}] \\ \rho_{61} &= \rho_{51} + \rho_{52}[\rho_{52} - \frac{\alpha_5}{2}\rho_{42}] \end{aligned} \tag{3.67}$$

Substituting for ρ_{52} from Eq. (3.30)

$$\rho_{52} = \rho_{42} + \rho_{41} - \alpha_4\rho_{31} \tag{3.68}$$

From Eq. (3.68) we can observe that the α_k recursion parameter and the number of previous coefficients required to be known are increasing in order, and it appears that a simple recursive algorithm based on the above approach is not possible. Note that although the new lattice structure does not appear to be order recursive, we can express a given order symmetric or antisymmetric lattice structure in a more conventional form.

In summary, we know that the split-Levinson algorithm is a viable replacement for the classical algorithm because of its computational efficiency. We have studied both autocorrelation and data (or lattice) realizations of the split-Levinson algorithm. An attempt to derive a recursive split lattice algorithm yielded a classical-like lattice structure, but it is not recursive in order. Further investigation is necessary in this direction. We now need to test the split-Levinson algorithms on some signal processing applications.

IV. APPLICATIONS OF THE SPLIT-LEVINSON ALGORITHM

In this chapter, we apply the split-Levinson algorithm in (1) the MA parameter estimation problem, and (2) the extended Prony method of spectral line estimation. Before we take up these two applications we examine the algorithm's usefulness if the original filter has coefficient symmetry, i.e., the impulse response of a linear phase FIR filter.

A. HANKEL AND TOEPLITZ MATRICES

In previous derivations we have assumed the FIR filter equation to be non-symmetric. Let us now investigate the problem where the filter equation is symmetric, i.e., of the form

$$\begin{aligned} y(n) = s(n) + a_1 s(n-1) + a_2 s(n-2) + \cdots \\ + a_{k-1} s(1) + a_k s(n-k) \end{aligned} \tag{4.1}$$

By definition, a symmetric polynomial is self-reciprocal, that is

$$a_k(z) = \hat{a}_k(z) = z^{-k} a_k(z^{-1}) \tag{4.2}$$

Therefore, from the Levinson algorithm, predictor polynomials are known to obey the recurrence relation

$$a_k(z) = a_{k-1}(z) + \rho_k z^{-1} \hat{a}_{k-1}(z) \tag{4.3}$$

and in our special case we have

$$\begin{aligned} a_k(z) &= a_{k-1}(z) + \rho_k z^{-1} \hat{a}_{k-1}(z) = \hat{a}(z) \\ &= (1 + \rho_k z^{-1}) a_{k-1}(z) \end{aligned} \tag{4.4}$$

In order to formulate a set of equations similar to the split-Levinson, it is necessary to derive the normal equations for our special case, and compare them to the standard equations, in order to develop the recursive algorithms. Since the predictor coefficients in a recursive algorithm produce estimates of s(n) as the algorithm steps through its recursive steps, denote this estimate as $\hat{s}(n)$. In vector form, we then have

$$\hat{s}(n|n-1) = -[s(n-1)\, s(n-2) \ldots s(0)] \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \ldots \\ a_2 \\ a_1 \\ 1 \end{bmatrix} \tag{4.5}$$

To derive the normal equations we must find the minimum mean squared error (MSE) from the equation for the error,

$$e(n) = s(n) - \hat{s}(n|n-1) \tag{4.6}$$

The minimum mean squared error is found by squaring the error term, and then differentiating the squared term with respect to the given a_k vector. Combining these two evolutions we have the following equations,

$$\begin{aligned} MSE &= J \\ &= E[e^2(n)] \\ &= E[(s(n) - (\hat{s}(n)|n-1))^2] \end{aligned} \tag{4.7}$$

To obtain the normal equations, we carry out the following steps:

$$\begin{aligned} \frac{\hat{c}J}{\hat{c}\mathbf{a}} &= 0 = -2E[s(n)\mathbf{s}_{k-1}^T] - 2E[\mathbf{s}_{k-1}^T \mathbf{s}_{k-1}] \\ E[s(k)\mathbf{s}_{k-1}\mathbf{s}_{k-1}^T]\mathbf{a} &= E[s(n)\mathbf{s}_{k-1}] \\ R^{(k-1)}\mathbf{a} &= \mathbf{r}_s^{(k-1)} \end{aligned} \tag{4.8}$$

From the split-Levinson recursion formulas we know that the singular symmetric polynomial, $P_k(z)$, is defined as the following,

$$P_k(z) = a_{k-1}(z) + z^{-1}\hat{a}_{k-1}(z) \tag{4.9}$$

Since our predictor polynomial is symmetric, it is a reasonable question to ask if symmetric polynomials also obey this recursive relationship. It is noted as an immediate consequence of the recursive problem, all of the preceding polynomials will also be symmetric. Therefore, we check the singular predictor polynomial recursion to see if it holds when the original polynomial is itself symmetric

$$
\begin{aligned}
a_k(z) &= a_{k-1}(z) + z^{-1}\hat{a}_{k-1}(z) \\
&= a_{k-1}(z) + z^{-1}z^{-k}a_{k-1}(z) \\
&= 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + z^{-(k-1)} + z^{-k}[1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + z^{-k}] \\
&= 1 + (1 + a_1)z^{-1} + (a_1 + a_1)z^{-2} + \ldots + z^{-k}
\end{aligned}
\tag{4.10}
$$

Now that we see that the Levinson recursive equation holds for a symmetric polynomial, we derive the recursive relationships for our polynomial using what is known from the split-Levinson equations. We have defined the symmetric polynomial $P_k(z)$ to be a normalized combination of a non symmetric polynomial, $a_k(z)$, and its reciprocal image, $\hat{a}_k(z)$ in the form of,

$$
\lambda_k P_k(z) = a_k(z) + \hat{a}_k(z) \tag{4.11}
$$

By direct substitution it is a trivial matter to show that this relationship also holds for a symmetric polynomial, $a_k(z)$.

In order to develop the recursion for the symmetric polynomial, it is necessary to express the desired linear predictor in terms of the previous two predictors. To this end use Eq. (4.10), to form $a_{k-1}(z)$, and substitute from Eq. (4.11) to perform this task.

$$
\begin{aligned}
a_k(z) &= a_{k-1}(z) + z^{-1}\hat{a}_{k-1}(z) \\
a_{k+1}(z) &= a_k(z) + z^{-1}\hat{a}_k(z) \\
&= a_k(z) + z^{-1}[\lambda_k a_k(z) - a_k(z)] \\
&= a_k(z)(1 - z^{-1}) + z^{-1}\lambda_k a_k(z)
\end{aligned}
\tag{4.12}
$$

Solving for $(1 - z^{-1})a_k(z)$, and forming the quantity $(1 - z^{-1})a_{k-1}(z)$ we have

$$
\begin{aligned}
(1 - z^{-1})a_k(z) &= a_{k+1}(z) - z^{-1}\lambda_k a_k(z) \\
(1 - z^{-1})a_{k-1}(z) &= a_k(z) - z^{-1}\lambda_{k-1}a_{k-1}(z)
\end{aligned}
\tag{4.13}
$$

These relationships will now allow us to form the three term recursion for the given symmetric polynomial from Eqs. (4.3), (4.11), (4.12), and (4.13)

$$
\begin{aligned}
a_k(z) &= a_{k-1}(z) + \rho_k \hat{a}_{k-1}(z) \\
a_{k+1}(z) &= (1 + z^{-1})a_k(z) - \alpha_k z^{-1}a_{k-1}(z)
\end{aligned}
\tag{4.14}
$$

where we have defined α_k

$$
\alpha_k = \lambda_{k-1}(2 - \lambda_k) \tag{4.15}
$$

From Eq. (4.14) we can see that the coefficient recursion formula is the same form as Eq. (3.29), and we can deduce that the split-Levinson algorithms will work equally well for symmetric polynomials that describe unknown filters as it does for polynomials that are not symmetric.

B. FIR MOVING AVERAGE PARAMETER ESTIMATION

If we consider an FIR filter with an input sequence given by $\mathbf{s}^T = [s(n)s(n-1)...s(n-m)]$, and an output y(n) given by

$$y(n) = \sum_{i=0}^{M} a_i\, s(n-i) \tag{4.16}$$

then we can develop the necessary equations to estimate the moving average parameters, and solve for the FIR filter coefficients. The algorithm to estimate the predictor coefficients can be defined as follows:

Let the three predictions, $\hat{y}_f^m(n)$, $\hat{s}_f^m(n)$, and $\hat{s}_b^m(n)$, represent the m-th order predictions of the forward estimate of y, the forward estimate of s, and the backward estimate of s respectively. [Ref. 6: p. 1]

$$\hat{y}_f^m(n) = \sum_{i=0}^{m} b_i\, s(n-i) \tag{4.17}$$

$$\hat{s}_f^m(n) = \sum_{i=0}^{m} c_i\, s(n-i) \tag{4.18}$$

$$\hat{s}_b^m(n-m) = \sum_{i=0}^{m} d_i\, s(n-m+i), \tag{4.19}$$

where the coefficient vectors are defined as,

$$\begin{aligned} \mathbf{b}^T &= [1\ -b_0\ -b_1\ ...\ -b_m] \\ \mathbf{c}^T &= [0\ 1\ -c_1\ ...\ -c_m] \\ \mathbf{d}^T &= [0\ -c_m\ -c_{m-1}\ ...\ -c_1\ ] \end{aligned} \tag{4.20}$$

Without going through all the details for the MA parmeter recursions, we can summarize the recursion formulas for the predictor coefficients as [Ref. 6: p. 2]

$$\mathbf{b}^m = \begin{pmatrix} \mathbf{b}^{m-1} \\ 0 \end{pmatrix} + K_y^m(\mathbf{d}^m)$$
$$\mathbf{c}^m = \begin{pmatrix} \mathbf{c}^{m-1} \\ 0 \end{pmatrix} + K_x^m \begin{pmatrix} 0 \\ \mathbf{d}^{m-1} \end{pmatrix} \tag{4.21}$$

Notice that the predictor vector $\mathbf{d}^m$ is not included in the preceding definitions since it is the reverse $\mathbf{c}^m$

If we examine Eq. (4.21) we see that the recursive relationship for $\mathbf{c}^m$ is a statement of the Levinson recursion, since K_x^m and K_y^m are the m-th order reflection coefficients. Therefore, we can apply the split-Levinson algorithm to solve for $\mathbf{c}^m$, form $\mathbf{d}^m$, and recursively determine $\mathbf{b}^m$. Finally, from the theory of Moving Average processes, $\mathbf{b}^m = \mathbf{a}^m$

The FORTRAN program MAV1, in Appendix D, uses a 25-th order FIR test filter to to generate a test sequence, and the results are given in Table 6 of Appendix A.

Figure 8 is a graphical comparison between the known test coefficients, shown by the solid curve, and the computed filter coefficients, shown by the dashed curve. The curves are fitted to the linear magnitudes of the coefficients by interpolating spline techniques.

C. EXTENDED PRONY METHOD

The estimation of the existence of sinusoids in the presence of noise is a common occurrence in signal processing applications. In this simulation, we will show that the split-Levinson algorithm can be directly implemented in the process to determine the approximate frequencies. The noise is zero mean, unit variance, and white in nature.

In this application of the split-Levinson algorithm, we attempt to approximately fit p exponentials to a data set of N samples. We have the constraint that $N > 2p$, and a least squares estimation procedure is used. We begin by defining the estimate of our set of data samples. [Ref. 7: p. 1404]

$$\hat{x} = \sum_{m=1}^{p} b_m z_m^n \quad n = 0,...,N-1 \tag{4.22}$$

where $b_m = A_m \exp(j\theta_m)/2$, and $z_m = \exp(j2\pi f_m \Delta t)$. The z_m 's are roots of unit modulus with arbitrary frequency and occur in complex conjugate pairs as long as $f_m \neq 0$ or $1/2\Delta t$ Therefore, in order to solve for the p sinusoids, we must solve for the roots of the following polynomial. [Ref. 7: p. 1406]

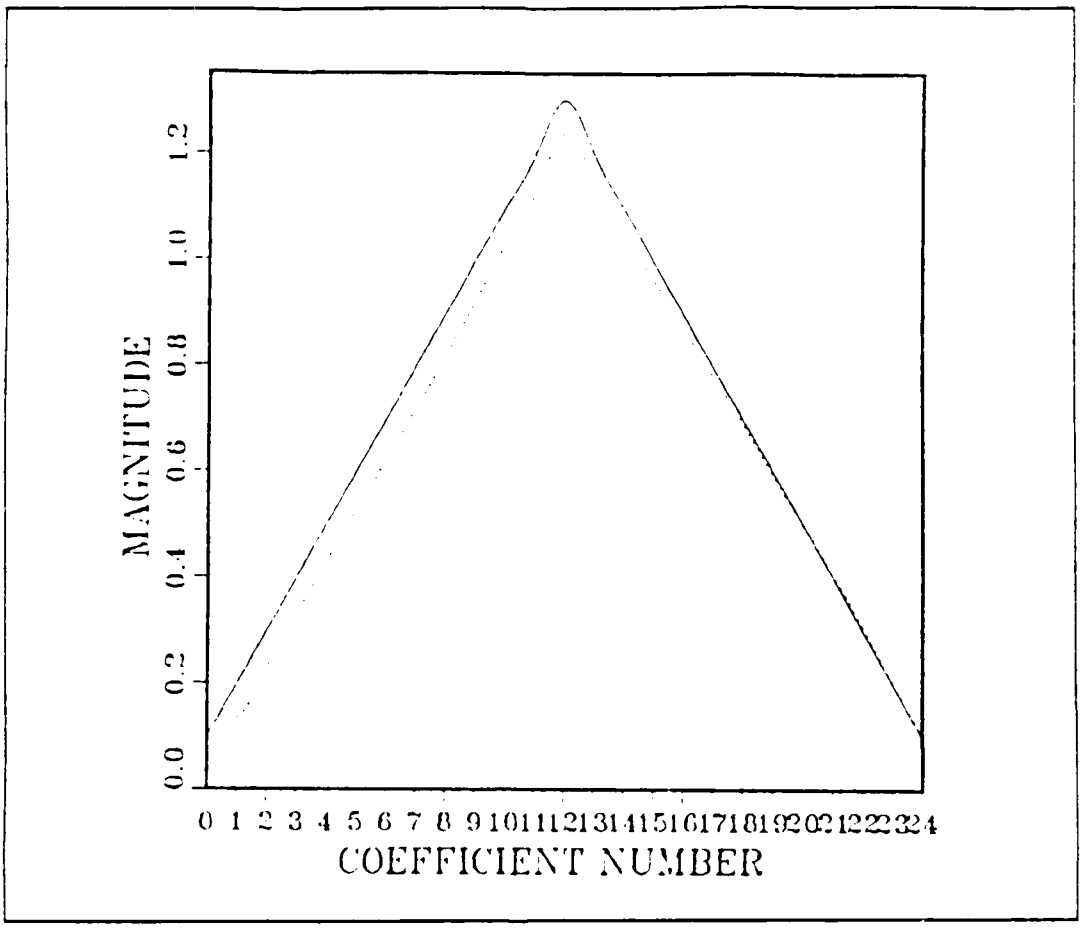


Figure 8. MA Coefficient Comparison.

$$\Psi(z) = \sum_{k=0}^{2p} a_k z^{2p-k} = 0 \tag{4.23}$$

The roots can be of unit moduius, and occur in complex conjugate conjugate pairs if we constrain the polynomial coefficients to be symmetric about the center element of the polynomial [Ref. 7: p. 1407]. This is an exact ocurrence in the symmetric and anti symmetric polynomials of the split-Levinson algorithm, as long as the order of the polynomial is even, that is

$$\mathbf{a}^T = \left[1 \; a_1 \; \ldots \; a_{p-1} \; \frac{a_p}{2} \right] \tag{4.24}$$

Note that the last element of the coefficient vector is a_p 2 rather than a_p because of the symmetry of the polynomial, and that symmetric polynomials only guarantee that if a root z_i occurs, then so does its reciprocal z_i^{-1} [Ref. 7: p. 1407].

The program EPRONY1, Appendix E, utilizes the split-Levinson algorithm to approximate the p order sinusoids to the given data set. A summary of the simulation cases studied are presented in Table 1, and the graphical results follow.

Table 1. SUMMARY OF TEST CASES

Case Number	Constants	Variables
1	Npts Test frequencies f_s	SNR(-10, 0, 10 dB)
2	Test frequencies f_s SNR	NPTS
3	Test frequencies f_s SNR Npts	Filter Order
4	Test frequencies f_s Npts SNR Filter Order	SNR (-10,0,10 dB)

1. Simulation Parameter Definitions

All simulation cases are done in the presence of white noise, and a minimum possible separation frequency for the input sinusoids was determined. All plots are sinusoid magnitude, in a linear scale, versus digital radian frequency, θ for $0 \leq \theta \leq \pi$.

2. Simulation Results

We begin the spectral line estimation simulations with all parameters fixed, and then selectively choose a parameter to vary and observe the effects of these changing parameters. We begin with two sinusoids of $f_1 = 50$ Hz, $f_2 = 75$ Hz, which yield $\theta_1 = 1.396$ radians, and $\theta_2 = 2.0944$ radians, respectively. The number of data points

(NPTS) is set at 1500, and the filter order (M) is chosen to be 4 indicating the presence of two sinusoids. Now, we chose to vary the signal-to-noise ratio (SNR) for 10 dB, 0 dB, and -10 dB, shown in Figure 9.

Figures 9 (a) and 9 (b) show that lowering the SNR from 10 dB, Figure 9 (a), to 0 dB, Figure 9 (b), causes the estimation to worsen, and both indicate low frequency estimation error. From both of these cases we can deduce the presence of two sinusoids, but in Figure 9 (c) it appears that only one sinusoid is present, and the low SNR has caused spectral estimation to fail. The conclusion for the first case is that, the better the SNR, the better the spectral estimate.

For the second case we selected NPTS as the variable parameter, held the filter order and sinusoid frequencies constant as before, and set the SNR at 0 dB. From Figures 10(a), (b), and (c) we clearly see that the better estimation occurs with NPTS = 1000, Figure 10(b), because of the equal amplitude and accurate frequency estimation as compared to Figures 10 (a) and 10 (c). All plots show low frequency error, but also suggests that simulations should be conducted with NPTS set to 1000-1500 points for the best results. This case provides the rationale for the value of NPTS for all other simulations.

In the third simulation all parameters are fixed as in the second case, and M is varied for 4,8, and 10. From Figures 11(a), (b), and (c) we can see that although there are only two sinusoids present, the estimation plots show M 2 sinusoids for all values of filter order. Each plot has frequency components in the vicinity of the actual frequencies, but they also give false indications of spectral lines. If we were making a decision on the number of frequencies based on the magnitude plots of Figures 11(b) and (c), then signifigant errors would be introduced. In this case it is obvious that unless the exact number of sinusoids present is known, then the estimation technique will fail.

In the final simulation we introduce two additional sinusoids, and examine the effects of SNR on spectral line estimation. The constants for this case are $f_1 = 35$ Hz, $f_2 = 85$ Hz, $f_3 = 105$ Hz, $f_4 = 175$ Hz, NPTS = 1500, M = 4, and $f_s = 525$ Hz. Digital frequencies are 0.419, 1.010, 1.496, and 2.094 radians per second respectively. As in case 1, SNR takes on the values of 10 dB, 0 dB, and -10 dB. In Figure 12(a), where the SNR is 10 dB, we get good results with near uniform amplitude estimation, and digital frequencies that are close for all frequencies. However, from Figures 12 (b) and 12 (c), we can see that the spectral line for f_3 is missing, and an errant line appears at approximately 2.8 radians. Both of the figures show unequal amplitude indications, but 3 of the 4

spectral lines are in close proximity to their actual values. From this case we draw the conclusion that the spectral line estimation performance deteriorates at low SNRs.

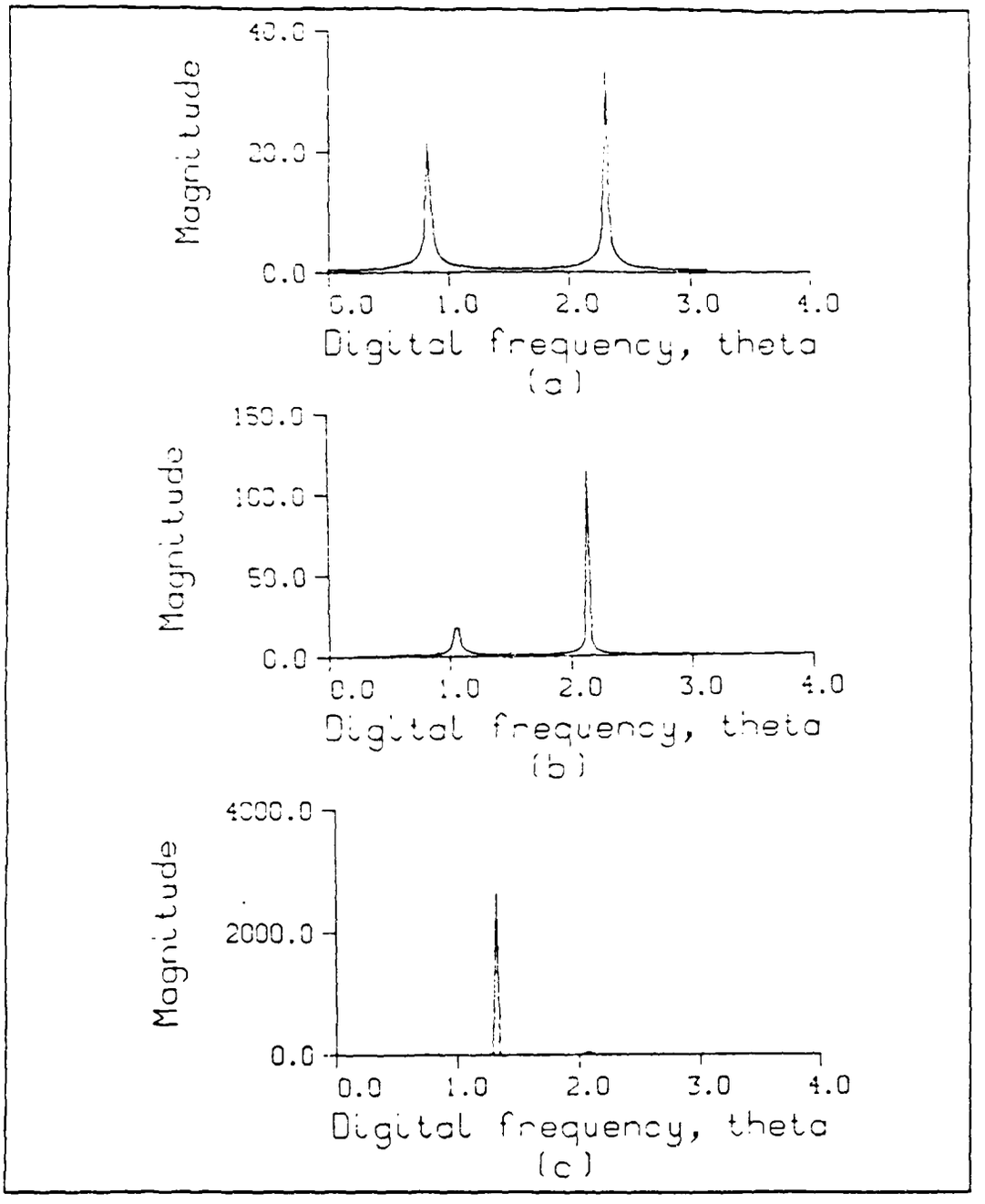


Figure 9. Spectral Estimation: Filter Order = 4; Data Record Length = 1500; SNRs: (a) 10 dB, (b) 0 dB, (c) -10 dB.

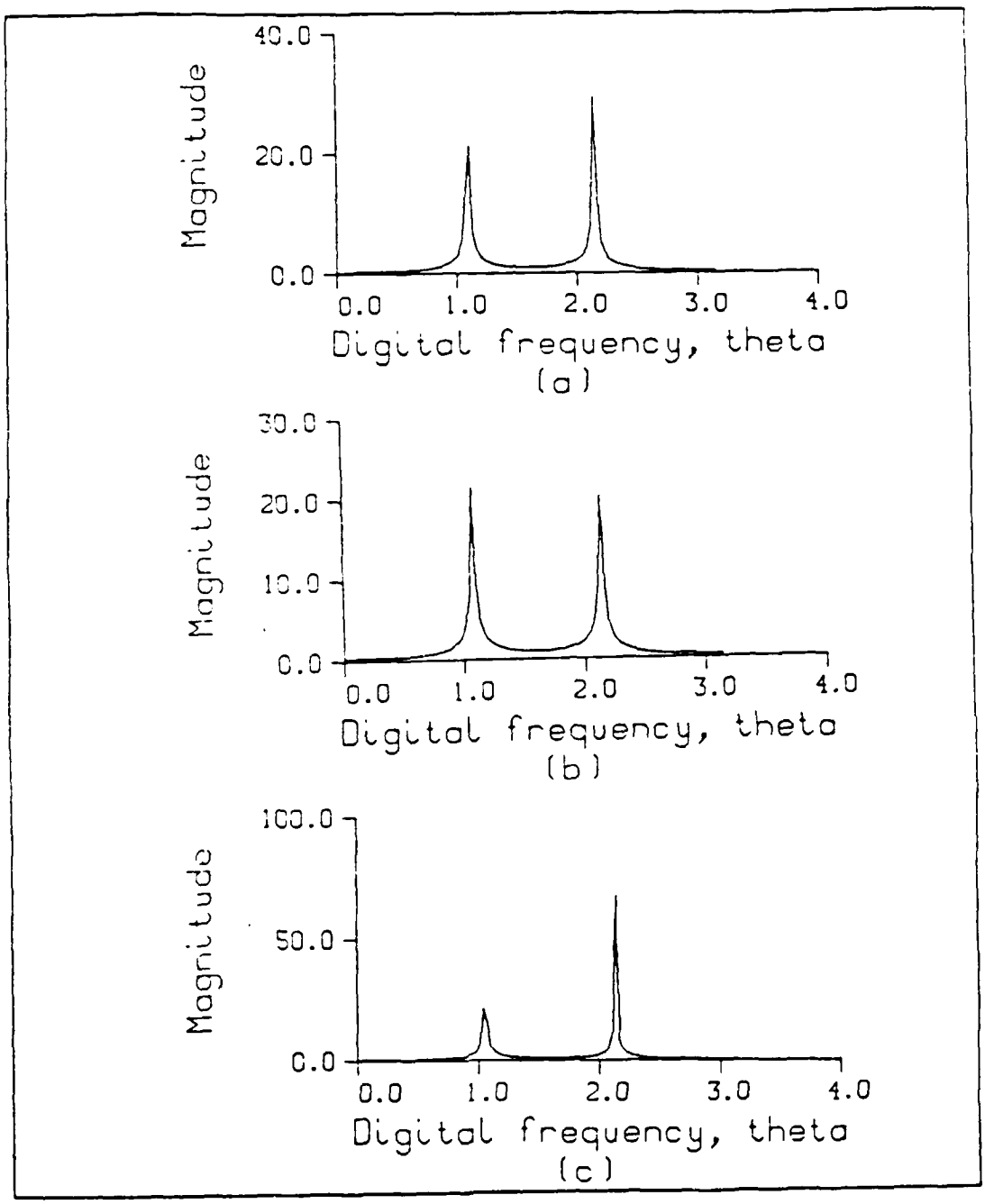


Figure 10. Spectral Estimation: Filter Order = 4: SNR = 0 dB, Data Record Lengths: (a) 500), (b) 1000, (c) 3000).

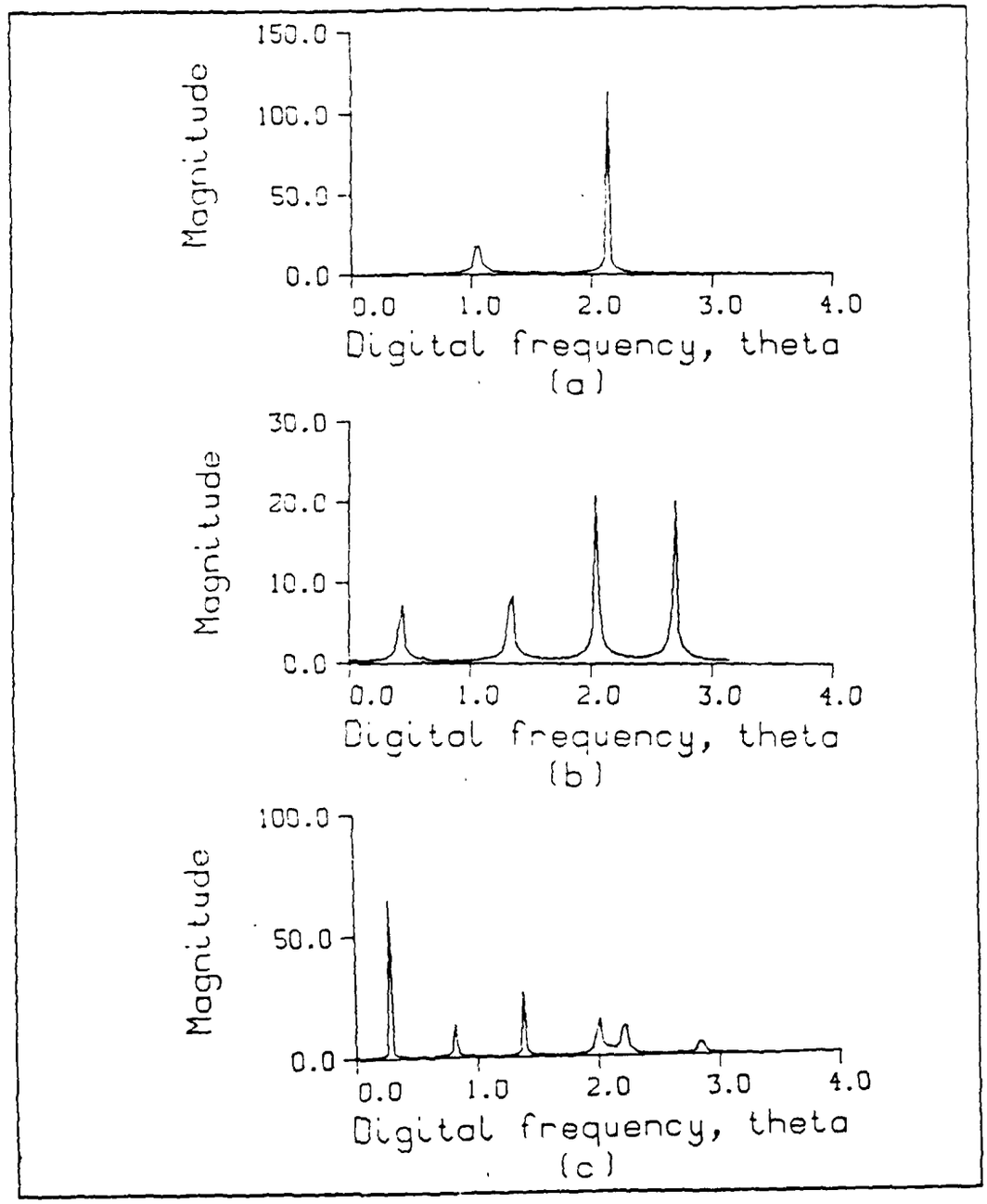


Figure 11. **Spectral Estimation: SNR = 0 dB; Data Record Length = 1500; Filter Orders: (a) 4, (b) 8, (c) 12.**

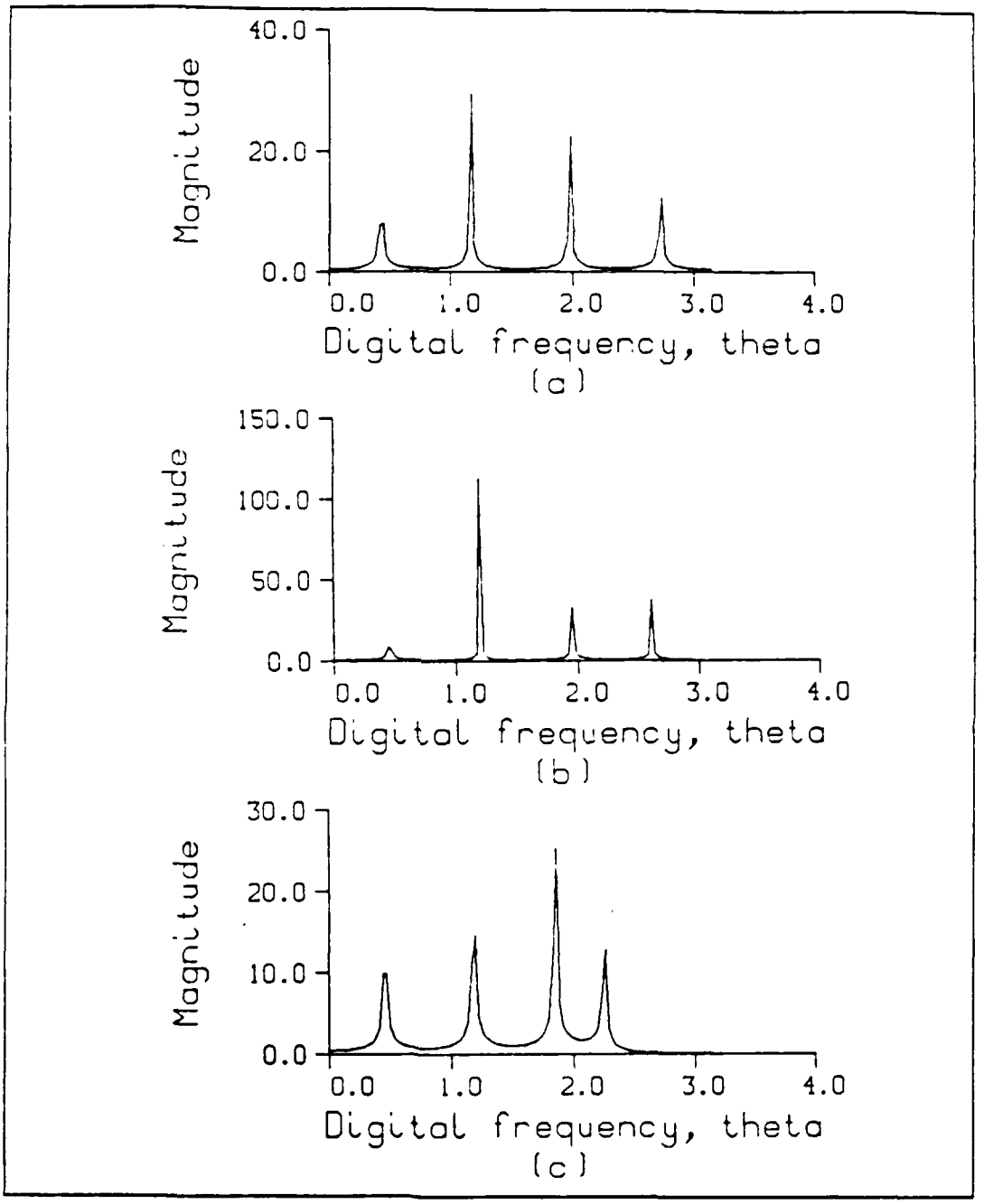


Figure 12. **Spectral Estimation (Four Sinusoids): Filter Order = 8; Data Record Length = 1500; SNRs: (a) 10 dB, (b) 0 dB, (c)-10 dB.**

Since the overall purpose of the simulation was to test the applicability of the split-Levinson algorithm to the test cases, then it has been shown that the split-Levinson will produce estimates for the respective cases. However, if we examine the accuracy of the low signal-to-noise ratio cases, we see that the new algorithm suffers a similar fate as the classical case in that it is difficult to accurately estimate the correct sinusoidal frequencies in the presence of noise.

D. CONCLUSIONS

The split-Levinson algorithm has been shown to be computationally more efficient than its classical counterpart. We can say that the application of the split-Levinson algorithms to practical applications in lieu of the Levinson algorithm can be advantageous, and the computational cost can be reduced significantly for large order systems. Additionally, the split-Levinson algorithms are applicable to problems where we want to model MA parameters, and perform spectral estimation using the Prony method.

We note the following restrictive areas for the new algorithm that could make it unsuitable for certain signal processing applications.

1. Non-recursive split-lattice algorithm.
2. Computational accuracy degradation when performing spectral estimation in low signal-to-noise cases.
3. Complexity of symmetric and antisymmetric lattice structures.

Since we have shown that the split-Levinson algorithm is a viable substitute for the Levinson recursion, it is reasonable to consider areas of this topic for further research. We know that the symmetric lattice structure can be expressed in a classical form for given filter orders, however, a recursive algorithm for this new structure has been elusive. We propose that the existing recursive algorithm for the singular predictor coefficients should be studied to see if a redefinition of equation parameters can extract a new recursive algorithm for the new symmetric lattice. Additionally, the algorithm's poor performance in low signal-to-noise ratio test cases of the extended Prony method is similar to the performance of the classical algorithm. Therefore, techniques to improve the classical algorithm's performance, as detailed in [Refs. 8,9], may be investigated for adaptation to the split-Levinson algorithm.

APPENDIX A. TABULAR SUMMARY OF ALGORITHMS

The tables given in this Appendix were taken from [Ref. 2: pp.648-674], and are presented for the convenience of the reader.

Table 2. THE LEVINSON ALGORITHM

Computation	Add	Mult
$a_{0,0} = 1, \quad \sigma_0 = c_0$		
For k = 1,2,...,n		
$\lambda_k = -\sum_{i=0}^{k-1} c_{k-i} a_{k-1,i}$	k-1	k-1
$a_{k,0} = 1, \quad \rho_k = \lambda_k / \sigma_{k-1}$		1
$\sigma_k = \sigma_{k-1} - \lambda_k \rho_k$	1	1
$a_{k,i} = a_{k-1,i} + \rho_k a_{k-1,k-i}$	k-1	k-1
$(i = 1,2,....,k-1)$		

Table 3. THE LEVINSON LATTICE ALGORITHM

Computation	Add	Mult
$f_0(t) = b_0(t) = s(t) \quad (0 \le t \le N-1)$ $\sigma_0 = \sum_{t=1}^{N-1} s(t)^2$	N-1	N
For k = 1,2,....n, $\lambda_k = -\sum_{t=1}^{N-k-2} f_{k-1}(t) b_{k-1}(t-1)$	N+k-3	N+k-2
$\rho_k = \lambda_k / \sigma_{k-1}$		1
$\sigma_k = \sigma_{k-1} - \lambda_k \rho_k$	1	1
$f_k(t) = f_{k-1}(t) + \rho_k b_{k-1}(t-1)$	N+k-2	N+k-1
$b_k(t) = \rho_k f_{k-1}(t) + b_{k-1}(t)$ $(t = 0,1,.....N+k-1)$	N+k-2	N+k-1

Table 4. THE SPLIT-LEVINSON ALGORITHM

Computation	Add	Mult
$p_{0,0} = 2,\ p_{1,0} = p_{1,1} = 1,\ \tau_0 = R_0,\ \rho_0 = 0$		
For k = 1,2,....n $k + 1 = 2t - \delta,$ with $\delta = 0$ or 1 $\tau_k = \sum_{i=0}^{t-1} (R_i + R_{k-i}) p_{k,i} + \delta c_t p_{k,t}$	t	$t - 1 + \delta$
$p_{k,0} = 1, \quad \alpha_k = \tau_k / \tau_{k-1}$		1
$\rho_k = 1 - \alpha_k / (1 + \rho_{k-1})$	2	1
$p_{k+1,i} = p_{k,i} + p_{k,i-1} - \alpha_k p_{k-1,i-1}$ $(i = 1,2,....,t)$	2t	t-1

Table 5. THE SPLIT LATTICE ALGORITHM

Computation	Add	Mult
$x_0(t) = 2s(t) \quad (0 \le t \le N-1)$ $x_1(t) = s(t) + s(t-1) \quad (0 \le t \le N)$	N-1	
$\rho_0 = 0\,, \quad \tau_0 = \sum_{t=0}^{N-1} s(t)^2$	N-1	N
For k = 1,2,....n $2\tau_k = \sum_{t=0}^{N+k-1} x_k(t)^2$	N + k-1	N + k
$\alpha_k = \tau_k / \tau_{k-1}$		1
$\rho_k = 1 - \alpha_k / (1 + \rho_{k-1})$	2	1
$x_{k+1}(t) = x_k(t) + x_k(t-1) - \alpha_k x_{k-1}(t)$ $(t = 0, \ldots, N+k)$	2(N + k-1)	N + k-1

Table 6. MOVING AVERAGE TEST RESULTS

Coefficient Index	Test Coefficients	Symmetric split-Levinson Coefficients
0	0.1	0.21
1	0.2	0.12
2	0.3	0.22
3	0.4	0.32
4	0.5	0.42
5	0.6	0.52
6	0.7	0.61
7	0.8	0.71
8	0.9	0.81
9	1.0	0.91
10	1.1	1.02
11	1.2	1.13
12	1.3	1.23
13	1.4	1.33
14	1.3	1.24
15	1.2	1.15
16	1.1	1.06
17	1.0	0.97
18	0.9	0.88
19	0.8	0.78
20	0.7	0.69
21	0.6	0.59
22	0.5	0.50
23	0.4	0.40
24	0.3	0.31
25	0.2	0.20
26	0.1	0.10

APPENDIX B. SPLIT-LEVINSON PROGRAMS

```
C       PROGRAM TO CALCULATE THE NTH ORDER FIR PREDICTOR FILTER USING
C       THE SYMMETRIC AND ASYMMETRIC SINGULAR PREDICTOR POLYNOMIALS,
C       THE SPLIT-LEVINSON RECURSION FORMULA, AND THE AUTOCORRELATION
C       FUNCTION OF THE INPUT SEQUENCE.
C                       VARIABLE DEFINITIONS
C        SIGMAN - N-TH DEGREE NORM OF THE FILTER.
C        KEVEN  - INTEGER VARIABLE USED TO CONTROL ACCESS TO
C                 SUBROUTINE EVEN WHEN THE INDEX VARIABLE K IS AN
C                 EVEN INTEGER.
C        LAMDA  - REAL VECTOR USED WHEN DEFINING THE SINGULAR
C        LAMDAS   PREDICTOR POLYNOMIAL (PK(Z)) IN TERMS OF THE
C                 NORMALIZED SYMMETRIC AND ANTISYMMETRIC PARTS OF
C                 THE PREDICTOR POLYNOMIAL AK(Z).
C                          LAMDA(K) = 1 + RHO(K)
C                         LAMDAS(K) = 1. - RHOS(K)
C        ALPHA  - REAL VECTOR USED TO SIMPLIFY THE THREE-TERM
C        ALPHAS   RECURRENCE RELATION FOR THE SINGULAR PREDICTOR
C                 POLYNOMIALS OF THE SAME TYPE.
C                    ALPHA(K) = LAMDA(K-1)*(2. - LAMDA(K)), OR
C                    ALPHA(K) = TAU(K)/TAU(K-1)
C                    ALPHAS(K) = TAUS(K)/TAUS(K-1)
C        KODD   - INTEGER VARIABLE USED TO CONTROL ACCESS TO THE
C                 SUBROUTINE ODD WHEN THE INDEX VARIABLE K IS AN
C                 ODD INTEGER.
C        TAU    - REAL VECTOR OF "MODIFIED NORM VALUES". THE
C        TAUS     VALUES ARE CALCULATED FROM A SUMMATION OF
C                 PRODUCT TERMS OF THE AUTOCORRELATION LAGS, AND
C                 THE COEFFICIENTS OF THE SINGULAR PREDICTOR
C                 POLYNOMIALS.
C        RHO    - REAL VECTOR OF REFLECTION COEFFICIENTS RHO(1),
C       RHOS      RHO(2),...,RHO(N).
C        N      - DESIRED ORDER OF THE PREDICTOR POLYNOMIAL.
C        C      - REAL VECTOR OF AUTOCORRELATION LAGS R(0),R(1),
C                 R(2),...,R(N)
C        P      - ARRAY OF SINGULAR PREDICTOR POLYNOMIAL
C        PS       COEFFICIENTS FROM P(0,0),...P(N+1,N).
C        A      - ARRAY OF PREDICTOR POLYNOMIAL COEFFICIENTS.
C        AS       EACH I-TH ROW OF THE ARRAY CONTAINS THE
C                 PREDICTOR POLYNOMIAL COEFFICIENTS FOR THE I-TH
C                 ORDER PREDICTOR POLYNOMIAL.
C        T      - INTEGER VARIABLE USED IN THE SUBROUTINE ODD
C                 IN THE COMPUTATION OF THE TAU(K)'S.
C        L      - DUMMY VARIABLE USED DURING THE CALCULATION
C                 OF THE SYMMETRIC SINGULAR PREDICTOR POLYNOMIAL
C                 COEFFICIENTS.
C                VARIABLE DECLARATIONS
```

```
      REAL C(0:100),P(0:100,0:100),TAU(0:100)
      REAL A(0:100,0:100),LAMDA(0:100),RHO(100),SIGMAN
      REAL PS(0:100,0:100),AS(0:100,0:100)
      REAL RHOS(100),ALPHAS(100),TAUS(0:100),SIGMAS
      REAL AR(0:30),W(0:5000),S(0:5000),SIGMA(0:100)
      REAL R(0:100),ALPHA(100),LAMDAS(0:100)
      REAL AA(0:100,0:100),GAM(50),LAMK,LAM(100)
      INTEGER M,LL,IX,T,KODD,KEVEN,L,N
         OPEN(UNIT=4,BLANK='ZERO')
C   INITIALIZE FILTER ORDER
         READ(4,100)N
         LL = N
         IX = 1913
          M = 3000
      WRITE(6,300)
      WRITE(6,400)N
      WRITE(6,450)M
         DO 1 I=0,M
         CALL GAUSS(IX,1.,0.,V)
         W(I) = V
 1       CONTINUE
         CALL INPUT(LL,W,AR,S,M)
C   INITIALIZE AUTOCORRELATION VECTOR C
         CALL ACORR(C,S,N+1,M)
         WRITE(6,2100)
C   INITIAL CONDITIONS FOR THE SYMMETRIC AND ASYMMETRIC PREDICTOR
C   POLYNOMIAL CALCULATIONS.
         P(0,0) = 2.
         P(1,1) = 1.
         TAU(0) = C(0)
         LAMDA(0) = 1.
         KODD = 1
         KEVEN = 2
         A(N,0) = 1.0
         PS(0,0) = 0.
         PS(1,1) = -1.
         TAUS(0) = C(0)
         LAMDAS(0) = 1.
         AS(N,0) = 1.0
         CALL LEV(C,GAM,N,AA,LAM,SIGMA)
         WRITE(6,1700)
         WRITE(6,1800)
         DO 20 J=1,N
         WRITE(6,1900)N,J,SIGMA(J),LAM(J),AA(N,J)
  20     CONTINUE
C   SYMMETRIC & ASYMMETRIC SPLIT-LEVINSON RECURSION
         WRITE(6,800)
         WRITE(6,850)
         WRITE(6,900)
         DO 99 K=1,N
```

```
          P(K,0) = 1.
          PS(K,0) = 1.
C    CALL STATEMENTS FOR EVEN OR ODD VALUES OF INDEX K
          IF(K.EQ.KODD)THEN
             CALL AODD(C,PS,K,N,TAUS,T)
             CALL ODD(C,P,K,N,TAU,KODD,T)
            ELSEIF(K.EQ.KEVEN)THEN
             CALL AEVEN(C,PS,K,N,TAUS,T)
             CALL EVEN(C,P,K,N,TAU,KEVEN,T)
          ENDIF
          ALPHA(K) = TAU(K)/TAU(K-1)
          ALPHAS(K) = TAUS(K)/TAUS(K-1)
C    LOOP TO CALCULATE SINGULAR PREDICTOR COEFFICIENTS
          DO 40 I=1,T
          P(K+1,I) = P(K,I) + P(K,I-1) - ALPHA(K)*P(K-1,I-1)
          PS(K+1,I) = PS(K,I) + PS(K,I-1) - ALPHAS(K)*PS(K-1,I-1)
C    DECISION PATH TO CALCULATE SYMMETRIC SINGULAR PREDICTOR COEFFICIENTS
          IF(I.LT.T .OR. I.EQ.K)GOTO 40
            L = K+1
            DO 50 J=I+1,K
            P(L,J) = P(L,L-J)
 50         PS(L,J) = -PS(L,L-J)
 40       CONTINUE
          LAMDA(K) = 2. - (ALPHA(K)/LAMDA(K-1))
          LAMDAS(K) = 2. - (ALPHAS(K)/LAMDAS(K-1))
C    REFLECTION COEFFICIENT CALCULATION
          RHO(K) = LAMDA(K) - 1.
          RHOS(K) = 1. - LAMDAS(K)
          WRITE(6,1000)K,RHO(K),RHOS(K),GAM(K)
 99       CONTINUE
C    CALCULATION OF N-TH ORDER NORM (SIGMAN) AND N-TH ORDER PREDICTOR
C    COEFFICIENTS, A(N,1),A(N,2),...,A(N,N)
          SIGMAN = LAMDA(N)*TAU(N)
          SIGMAS = LAMDAS(N)*TAUS(N)
          WRITE(6,1100)
          WRITE(6,600)
          WRITE(6,1200)
          DO 60 I=1,N
          A(N,I) = A(N,I-1) + P(N+1,I) - LAMDA(N)*P(N,I-1)
          AS(N,I) = AS(N,I-1) + PS(N+1,I) - LAMDAS(N)*PS(N,I-1)
          WRITE(6,1300)I,TAU(I),TAUS(I),ALPHA(I),ALPHAS(I),P(I+1,I),
     +PS(I+1,I),A(N,I),AS(N,I)
 60       CONTINUE
 100      FORMAT(I2)
 200      FORMAT(F12.6)
 300      FORMAT('1')
 400      FORMAT('  FILTER ORDER = ',I3)
 450      FORMAT('-',' NUMBER OF SAMPLE POINTS  = ',I5)
 600      FORMAT('-',103X,'FILTER COEFFICIENTS')
 700      FORMAT(5X,I3,11X,F10.4,21X,F10.4)
```

```
  800    FORMAT('-',21X,'REFLECTION COEFFICIENTS')
  850    FORMAT('-',25X,'SPLIT-LEVINSON')
  900    FORMAT(5X,'INDEX',8X,'SYMMETRIC',9X,'ANTISYMMETRIC',9X,
     +'LEVINSON')
 1000    FORMAT(5X,I3,10X,F12.6,12X,F12.6,12X,F12.6)
 1100    FORMAT('-',5X,'FILTER PARAMETERS FROM SPLIT LEVINSON RECURSION')
 1200    FORMAT(5X,'INDEX',6X,'TAU(K)',4X,'TAU*(K)',4X,'ALPHA(K)',4X,
     +'ALPHA*(K)',4X,'P(K)',4X,'P*(K)',6X,'SYMMETRIC',6X,'ASYMMETRIC')
 1300    FORMAT(5X,I3,6X,F12.6,3X,F12.6,3X,F12.6,4X,F12.6,3X,F12.6,2X,
     +F12.6,2X,F10.4,5X,F10.4)
 1700    FORMAT('-',5X,'FILTER PARAMETERS FROM LEVINSON RECURSION')
 1800   FORMAT('-',8x,'INDEX',8X,'SIGMA(K)',5X,'LAMDA(K)',8X,'FILTER
     +COEFFICIENTS')
 1900    FORMAT(8X,I2,I3,7X,F12.6,6X,F12.6,12X,F12.6)
 2000    FORMAT(5X,'SPLIT-LEVINSON RECURSION CALCULATIONS')
 2100    FORMAT(5X,'INDEX',5X,'KNOWN COEFFICIENTS',5X,'AUTOCORRELATION
     +FUNCTION C(K)')
 2200    FORMAT(2X,I3,4X,F12.6)
 2300    FORMAT(5X,I3,40X,F12.6)
      WRITE(6,300)
         END
         SUBROUTINE AODD(C,PS,K,N,TAUS,T)
C        THIS SUBROUTINE CALCULATES THE ANTISYMMETRIC "MODIFIED
C        NORMS" WHEN THE INDEX K IS AN ODD INTEGER.
      REAL C(0:100),PS(0:100,0:100),TAUS(0:100)
      INTEGER T
      T = (K-1)/2
      TEMP = 0.
      DO 5 I=0,T
  5   TEMP = TEMP + (C(I) - C(K-I))*PS(K,I)
      TAUS(K) = TEMP
      RETURN
      END
         SUBROUTINE ODD(C,P,K,N,TAU,KODD,T)
C        THIS SUBROUTINE CALCULATES THE "MODIFIED NORMS" ( TAU(K)'S)
C        WHEN THE INDEX K IS AN ODD INTEGER.
      REAL C(0:100),P(0:100,0:100),TAU(0:100)
      INTEGER T
      T = (K+1)/2
      TEMP = 0.
      DO 15 I=0,T-1
 15   TEMP = TEMP + (C(I) + C(K-I))*P(K,I)
      TAU(K) = TEMP
      KODD = KODD + 2
      RETURN
      END
         SUBROUTINE AEVEN(C,PS,K,N,TAUS,T)
C        SUBROUTINE CALCULATES THE VALUE OF THE ANTISYMMETRIC
C        "MODIFIED NORMS" (TAUS(K)'S) WHEN THE INDEX K IS AN EVEN
C        INTEGER.
```

```
      REAL C(0:100),PS(0:100,0:100),TAUS(0:100)
      INTEGER T
      T = K/2
      TEMP= 0.
      PS(K,T) = 0.
      DO 25 I=0,T-1
   25 TEMP = TEMP + (C(I) - C(K-I))*PS(K,I)
      TAUS(K) = TEMP
      RETURN
      END
         SUBROUTINE EVEN(C,P,K,N,TAU,KEVEN,T)
C        SUBROUTINE CALCULATES THE VALUE OF THE "MODIFIED NORMS"
C        (TAU(K)'S) WHEN THE INDEX K IS AN EVEN INTEGER.
      REAL C(0:100),P(0:100,0:100),TAU(0:100)
      INTEGER T
      T = K/2
      TEMP= 0.
      DO 35 I=0,T-1
   35 TEMP = TEMP + (C(I) + C(K-I))*P(K,I)
      TAU(K) = TEMP + C(T)*P(K,T)
      KEVEN = KEVEN + 2
      RETURN
      END
      SUBROUTINE INPUT(LL,W,AR,S,M)
C     SUBROUTINE TO GENERATE THE INPUT SEQUENCE FROM A GIVEN FIR
C     FILTER AND ZERO MEAN, UNIT VARIANCE WHITE NOISE.
      REAL W(0:M),AR(0:LL),S(0:M)
C   CALCULATE INPUT SEQUENCE VALUES BETWEEN 0 AND FILTER ORDER.
C     TEMP = 0.
      S(0) = W(0)
      DO 45 K=1,M
      S(K) = W(K) -0.6*W(K-1) + 0.8*S(K-1)
   45 CONTINUE
      RETURN
      END
      SUBROUTINE ACORR(C,S,NLAG,M)
C     SUBROUTINE TO CALCULATE THE AUTO CORRELATION FUNCTION OF THE
C     GIVEN INPUT SEQUENCE.
      REAL C(0:100),S(0:5000)
      INTEGER T
      DO 5 I=0,NLAG
      TEMP = 0.
      DO 15 T=0,M-1-I
      TEMP = TEMP + S(T)*S(T+I)
   15 CONTINUE
      C(I) = TEMP*(1./FLOAT (M-1-I))
   5  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE LEV(C,GAM,N,AA,LAM,SIGMA)
C     SUBROUTINE TO CALCULATE THE PREDICTOR FILTER COEFFICIENTS
C     USING THE LEVINSON RECURSION.
      REAL AA(0:100,0:100),C(0:N+2),GAM(N),LAM(100)
      REAL LAMK,SIGMA(0:100)
C     INITIAL CONDITIONS
      AA(0,0) = 1.
      SIGMA(0) = C(0)
C     COMPUTE LAM(K), RHO(K); UPDATE SIGMA(K) FOR THE NEXT
C     ITERATION.
      DO 10 K=1,N
      LAMK = 0.
      AA(K,0) = 1.0
      DO 20 I=0,K-1
       LAMK = LAMK - C(K-I)*AA(K-1,I)
       LAM(K) = LAMK
   20 CONTINUE
      GAM(K) = LAM(K)/SIGMA(K-1)
      SIGMA(K) = SIGMA(K-1) - LAM(K)*GAM(K)
C     COMPUTE THE PREDICTOR FILTER COEFFICIENTS
      IF(K .EQ. 1)THEN
       AA(1,1) = GAM(K)
      ELSE
       DO 30 I=1,K-1
        AA(K,I) = AA(K-1,I) + GAM(K)*AA(K-1,K-I)
   30  CONTINUE
       AA(K,K) = GAM(K)
      ENDIF
   10 CONTINUE
      RETURN
      END
```

APPENDIX C. SPLIT LATTICE ALGORITHMS

```
C       PROGRAM TO CALCULATE THE NTH ORDER LATTICE REFLECTION
C       COEFFICIENTS FROM A GIVEN SEQUENCE USING THE SYMMETRIC ERROR
C       VECTOR, THE ASYMMTRIC ERROR VECTOR, OR THE FORWARD AND
C       BACKWARD ERROR VECTORS. VARIABLES DEFINED IN PREVIOUS APPENDICES
C       ARE NOT REDEFINED.
C   ****** THIS PROGRAM REQUIRES SUBROUTINE INPUT FROM APPENDIX A ********
C                       VARIABLE DEFINITIONS
c         SIG     - N-TH DEGREE NORM OF THE FILTER.
C         GAM     - VECTOR OF REFLECTION COEFFICIENTS CALCULATED BY
C                   THE LEVINSON RECURSION.
C         LAM     - REAL VARIABLE USED WHEN CALCULATING THE REFLECTION
C                   COEFFICIENTS FROM THE LEVINSON RECURSION
C                   THE REFLECTION COEFFICIENT IN TERMS
C                   OF THE FILTER NORM IS GIVEN BY:
C                         RHO(K) = LAM/SIG
C         TAU     - REAL VECTOR OF "MODIFIED NORM VALUES". THE
C                   VALUES ARE CALCULATED FROM A SUMMATION OF
C                   PRODUCT TERMS OF THE SYMMETRIC OR ASYMMETRIC
C                   PREDICTION ERROR SEQUENCES.
C         A       - ARRAY OF PREDICTOR POLYNOMIAL COEFFICIENTS.
C         AS        EACH I-TH ROW OF THE ARRAY CONTAINS THE
C         AL        PREDICTOR POLYNOMIAL COEFFICIENTS FOR THE I-TH
C                   ORDER PREDICTOR POLYNOMIAL.
C         AR      - VECTOR OF COEFFICIENTS FROM THE KNOWN TEST FILTER.
C         XO      - SYMMETRIC OR ASSYMMETRIC PREDICTION ERROR VECTOR
C                   FOR THE (K-1) STAGE OF THE LATTICE FILTER.
C         LL      - DESIRED LATTICE FILTER ORDER.
C         X1      - SYMMETRIC OR ASYMMETRIC PREDICTION ERROR VECTOR
C                   FOR THE K-TH STAGE OF THE LATTICE FILTER.
C         AT      - TEMP STORAGE FOR THE PREDICTION ERROR VECTOR WHILE
C                   COMPUTING THE (K+1) STAGE PREDICTION ERROR VECTOR.
C         FT      - SHIFTED FORWARD PREDICTION ERROR VECTOR.
C         BT      - SHIFTED BACKWARD PREDICTION EROR VECTOR.
C         M       - DESIRED ORDER OF THE PREDICTOR POLYNOMIAL.
C         T       - INTEGER VARIABLE USED IN THE PROGRAM.
C         W       - WHITE NOISE SEQUENCE VECTOR.
C         S       - INPUT SEQUENCE VECTOR
C         F       - FORWARD PREDICTION ERROR VECTOR.
C         B       - BACKWARD PREDICTION ERROR VECTOR.
C                   VARIABLE DECLARATIONS
          REAL AR(30),W(0:5000),S(0:5000),RHO(100)
          REAL A(0:100,0:100),GAM(20),RHOS(100),AS(0:100,0:100)
          REAL ALPHA,X1(0:5000),XO(0:5000),AT(0:5000),AL(0:100,0:100)
          INTEGER M,LL,IX,T,L,N
          OPEN(UNIT=4,BLANK='ZERO')
C   INITIALIZE FILTER ORDER
```

```
      READ(4,100)M
C   INITIAL CONDITIONS FOR INPUT SEQUENCE GENERATOR
      LL = M
      N = 5000
      IX = 1913
      WRITE(6,200)
      WRITE(6,300)M
      WRITE(6,400)N
      WRITE(6,500)
      WRITE(6,600)
C   LOOP TO GENERATE WHITE NOISE SEQUENCE AND TO READ TEST COEFFICIENTS.
      DO 1 I=0,N
       CALL GAUSS(IX,1.,0.,V)
       W(I) = V
 1    CONTINUE
C   CALL STATEMENT TO GENERATE INPUT SEQUENCE
      CALL INPUT(LL,W,AR,S,N)
C     CALL STATEMENTS FOR SYMMETRIC, ASYMMETRIC, AND LEVINSON LATTICE
C     RECURSION SUBROUTINES
      CALL SLAT(S,M,N,RHO,ALPHA,X1,AT,XO)
      CALL ASLAT(S,M,N,RHOS,ALPHA,X1,AT,XO)
      CALL LEV(N,GAM,M,S)
      WRITE(6,700)
      WRITE(6,800)
      DO 80 K=1,M
      WRITE(6,900)K,RHO(K),RHOS(K),GAM(K)
 80   CONTINUE
      DO 90 K=1,M
       A(K,0)=1.
       AS(K,0)=1.
       AL(K,0)=1.
       IF(K .EQ. 1)THEN
        A(1,1) = RHO(K)
        AS(1,1) = RHOS(K)
        AL(1,1) = GAM(K)
       ELSE
        DO 95 I=1,K-1
         A(K,I) = A(K-1,I) + RHO(K)*A(K-1,K-I)
         AS(K,I) = AS(K-1,I) + RHOS(K)*AS(K-1,K-I)
         AL(K,I) = AL(K-1,I) + GAM(K)*AL(K-1,K-I)
 95      CONTINUE
        A(K,K) = RHO(K)
        AS(K,K) = RHOS(K)
        AL(K,K) = GAM(K)
       ENDIF
 90   CONTINUE
      WRITE(6,1000)
      WRITE(6,1100)
      WRITE(6,1200)
      DO 96 K=1,M
```

```
      WRITE(6,1300)M,K,AL(M,K),A(M,K),AS(M,K)
 96   CONTINUE
 100  FORMAT(I2)
 200  FORMAT('1')
 300  FORMAT('  FILTER ORDER = ',I3)
 400  FORMAT(' ','  NUMBER OF SAMPLE POINTS = ',I5)
 500  FORMAT('-',10X,'KNOWN FILTER COEFFICIENTS')
 600  FORMAT('-',8X,'INDEX',10X,'FILTER COEFFICIENTS')
 700  FORMAT('-',10X,'REFLECTION COEFFICIENTS')
 800  FORMAT('-',5X,' INDEX ',3X,'SYMMETRIC',9X,'ANTISYMMETRIC'
     +,9X,' LEVINSON ')
 900  FORMAT('-',6X,I3,6X,F8.4,12X,F8.4,12X,F8.4)
1000  FORMAT('-',15X,' FILTER COEFFICIENTS ')
1100  FORMAT('-',20X,' LEVINSON ',12X,' SPLIT-LEVINSON ')
1200  FORMAT(5X,' INDEX ',26X,' SYMMETRIC ',4X,' ASYMMETRIC ')
1300  FORMAT(' ',6X,2I2,10X,F8.4,11X,F8.4,7X,F8.4)
      WRITE(6,200)
      END
      SUBROUTINE SLAT(S,M,N,RHO,ALPHA,X1,AT,XO)
C     SUBROUTINE TO COMPUTE THE LATTICE REFLECTION COEFFICIENTS
C     USING THE SYMMETRIC PREDICTION ERROR VECTOR.
      REAL X1(0:M+N),XO(0:M+N),RHO(M),S(0:N),ALPHA
      REAL AT(0:M+N),A(100,100)
C     INITIAL CONDITIONS
      INTEGER T
      RRHO = 0.
      TAU = 0.
C     INITIALIZE THE PREDICTION ERROR VECTORS FOR THE ZERO AND 1ST
C     STAGES OF THE LATTICE. INITIALIZE THE ZERO STAGE MODIFIED NORM
      DO 10 T=0,N-1
      XO(T) = 2.*S(T)
      TAU = TAU + S(T)**2
 10   CONTINUE
      DO 20 T=0,N
       IF(T.EQ.N)S(T) = 0.
       IF(T.EQ.0)THEN
        X1(T) = S(T)
       ELSE
        X1(T) = S(T) + S(T-1)
       ENDIF
 20   CONTINUE
C     LOOP TO COMPUTE THE REFLECTION COEFFICIENTS
      DO 101 K=1,M
      TTAU = TAU
C      STORE TAU(K-1), AND COMPUTE TAU(K).
      TAU = 0.
      DO 30 T=0,N+K-1
       TAU = TAU + X1(T)**2
 30   CONTINUE
```

```
         TAU = TAU/2.
C      COMPUTE ALPHA(K), RHO(K); STORE TAU(K) AND RHO(K) FOR
C      NEXT ITERATION.
         ALPHA = TAU/TTAU
         TTAU = TAU
         RHO(K) = 1. - (ALPHA/(1. + RRHO))
         RRHO = RHO(K)
C      LOOP TO COMPUTE THE (K+1) PREDICTION ERROR VECTOR.
         DO 40 T=0,N+K
          IF(T .EQ. 0)THEN
           AT(T) = X1(T)
          ELSEIF(T.EQ.N+K)THEN
           AT(T) = X1(T-1)
           ELSE
            AT(T) = X1(T) + X1(T-1) - ALPHA*XO(T-1)
           ENDIF
  40     CONTINUE
C      LOOPS TO UPDATE PREDICTION ERROR VECTORS FOR NEXT ITERATION.
C         (SHIFT X1 INTO XO AND AT INTO X1)
         DO 50 T=0,N+K-1
          XO(T) = X1(T)
  50     CONTINUE
         DO 60 T=0,N+K
          X1(T) = AT(T)
  60     CONTINUE
  101    CONTINUE
         RETURN
         END
         SUBROUTINE ASLAT(S,M,N,RHOS,ALPHA,X1,AT,XO)
C        SUBROUTINE TO COMPUTE THE LATTICE REFLECTION COEFFICIENTS
C        USING THE ASYMMETRIC PREDICTION ERROR VECTOR.
         REAL X1(0:M+N),XO(0:M+N),RHOS(M),S(0:N),ALPHA
         REAL AT(0:M+N)
         INTEGER T
C     INITIAL CONDITIONS
         RRHO = 0.
         TAU = 0.
C    INITIALIZE THE PREDICTION ERROR VECTORS FOR THE ZERO AND 1ST
C    STAGES OF THE LATTICE. INITIALIZE THE ZERO STAGE MODIFIED NORM
         DO 10 T=0,N-1
          XO(T) = 0.
          TAU = TAU + S(T)**2
  10     CONTINUE
         DO 20 T=0,N
          IF(T.EQ.N)X1(T) = -S(T-1)
          IF(T.EQ.0)THEN
           X1(T) = S(T)
          ELSE
           X1(T) = S(T) - S(T-1)
          ENDIF
```

```
  20      CONTINUE
C     LOOP TO COMPUTE THE REFLECTION COEFFICIENTS
        DO 101 K=1,M
C      STORE TAU(K-1), AND COMPUTE TAU(K).
         TTAU = TAU
         TAU = 0.
        DO 30 T=0,N+K-1
         TAU = TAU + X1(T)**2
  30    CONTINUE
        TAU = TAU/2.
C      COMPUTE ALPHA(K), RHO(K); STORE TAU(K) AND RHO(K) FOR
C      NEXT ITERATION.
        ALPHA = TAU/TTAU
        TTAU = TAU
        RHOS(K) = (ALPHA/(1. - RRHO)) - 1.
        RRHO = RHOS(K)
C      LOOP TO COMPUTE THE (K+1) PREDICTION ERROR VECTOR.
        DO 40 T=0,N+K
         IF(T .EQ. 0)THEN
          AT(T) = X1(T)
         ELSEIF(T.EQ.N+K)THEN
          AT(T) = X1(T-1)
          ELSE
           AT(T) = X1(T) + X1(T-1) - ALPHA*XO(T-1)
          ENDIF
  40    CONTINUE
C      LOOPS TO UPDATE PREDICTION ERROR VECTORS FOR NEXT ITERATION.
C         (SHIFT X1 INTO XO AND AT INTO X1)
        DO 50 T=0,N+K-1
         XO(T) = X1(T)
  50    CONTINUE
        DO 60 T=0,N+K
         X1(T) = AT(T)
  60    CONTINUE
  101   CONTINUE
        RETURN
        END
        SUBROUTINE LEV(N,GAM,M,S)
C       SUBROUTINE TO COMPUTE THE REFLECTION COEFFICIENTS FOR AN
C       N-TH ORDER LATTICE FILTER FROM THE FORWARD AND BACKWARD
C       PREDICTION ERROR VECTORS.
        REAL F(0:5100),B(0:5100),FT(0:5100),BT(0:5100),GAM(20)
        REAL LAM,SIG,S(0:N)
        INTEGER T
C      INITIAL CONDITIONS; INITIALIZE FORWARD AND BACKWARD PREDICTION
C      ERROR VECTORS.
        SIG = 0.
        DO 10 I=0,N-1
         F(I) =S(I)
         B(I) = S(I)
```

```
          FT(I) = S(I)
          SIG = SIG + S(I)**2
   10     CONTINUE
C     LOOP TO COMPUTE THE REFLECTION COEFFICIENTS
          DO 20 K=1,M
C     FORM SHIFTED ERROR VECTORS
          DO 30 T=1,N+K-1
           BT(T) = B(T-1)
   30     CONTINUE
          BT(0) = 0.
          FT(N+K-1) = 0.
          LAM = 0.
C     COMPUTE LAM(K), GAM(K); UPDATE K-TH ERROR NORM AND
C     STORE FOR NEXT ITERATION.
          DO 40 T=1,N+K-2
           LAM = LAM - FT(T)*BT(T)
   40     CONTINUE
          GAM(K) = LAM/SIG
          IF(K .EQ. M)GOTO 20
          SIG = SIG - LAM*GAM(K)
C     COMPUTE (K+1) FORWARD AND BACKWARD PREDICTION ERRORS AND SHIFT
C     INTO TEMPORARY VECTORS FOR NEXT ITERATION.
          DO 50 T=0,N+K-1
           F(T) = FT(T) + GAM(K)*BT(T)
           B(T) = GAM(K)*FT(T) + BT(T)
   50     CONTINUE
          DO 60 T=0,N+K-1
           FT(T) = F(T)
           BT(T) = B(T)
   60     CONTINUE
   20     CONTINUE
          RETURN
          END
```

APPENDIX D. MA PREDICTOR COEFFICIENT PROGRAM

```
C       THIS PROGRAM IS TO COMPUTE THE FIR FILTER COEFFICIENTS
C       USING THE SPLIT-LEVINSON ALGORITHM, AND AN AUTOREGRESSIVE
C       MOVING AVERAGE PROCESS. VARIABLES DEFINED IN PREVIOUS
C       APPENDICES ARE NOT REDEFINED.
C  ***** THIS PROGRAM REQUIRES THE SUBROUTINES ODD, AND EVEN
C             FROM APPENDIX A *****
C                    VARIABLE DEFINITIONS
C        ENORM  - PREDICTOR COEFFICIENT ERROR NORM.
C                 ENORM = SQRT((A(0)-AA(0))**2 +...+ (A(N)-AA(N))**2)
C        NSTRT  - NUMBER OF POINTS OF INPUT SEQUENCE TO START.
C        NEND   - NUMBER OF POINTS OF INPUT SEQUENCE AT END OF PROGRAM.
C        DELX   - ERROR VECTOR.
C        DELY   - ERROR VECTOR.
C        NPTS   - NUMBER OF INPUT DATA POINTS (0,1,...,NPTS).
C        EXB    - BACKWARD PREDICTION ERROR POWER OF X.
C        RXY    - VECTOR OF X AND Y CROSSCORRLATION ELEMENTS
C        AA     - VECTOR OF CALCULATED PREDICTOR COEFFICIENTS.
C        NX     - INDEX FOR X-AXIS OF PLOTTING ROUTINE.
C        EN     - VECTOR OF PREDICTOR COEFFICIENT NORMS.
C        EX     - FORWARD PREDICTION ERROR POWER OF X.
C        EY     - FORWARD PREDICTION ERROR POWER OF Y.
C        KY     - Y REFLECTION COEFFICIENT.
C        KX     - X REFLECTION COEFFICIENT.
C        LL     - FILTER ORDER VARIABLE USED IN SUBROUTINE CORR.
C        IX     - INTEGER SEED NUMBER USED BY IMSL SUBROUTINE GAUSS.
C        RX     - X AUTOCORRELATION VECTOR.
C        RY     - Y AUTOCORRELATION VECTOR.
C        B      - MA COEFFICIENT VECTOR.
C        C      - MA COEFFICIENT VECTOR.
C        D      - MA COEFFICIENT VECTOR.
C        T      - INTEGER VARIABLE USED IN THE SUBROUTINE ODD
C                 IN THE COMPUTATION OF THE TAU(K)'S.
C        L      - DUMMY VARIABLE USED DURING THE CALCULATION
C                 OF THE SYMMETRIC SINGULAR PREDICTOR POLYNOMIAL
C                 COEFFICIENTS.
C        X      - INPUT WHITE NOISE VECTOR.
C        M      - INDEXING VARIABLE USED IN FIR FILTER COEFFICIENT
C                 RECURSION (M=1,2,...,N).
C        Y      - OUTPUT SEQUENCE FROM FIR TEST FILTER.
C           VARIABLE DECLARATIONS
      REAL P(0:100,0:100),TAU(0:100),C(0:50),B(0:50),EN(200)
      REAL A(0:100,0:100),LAMDA(0:100),X(0:10000),D(0:50)
      REAL AR(0:30),Y(0:10000),EY(0:50),EX(0:50),KX(50)
      REAL DELX(0:50),DELY(0:50),EXB(0:50),KY(50),XX(200)
      REAL RX(0:100),ALPHA(100),RY(0:2),RXY(0:100),AA(0:100)
```

```
      INTEGER M,LL,IX,T,KODD,KEVEN,L,N,NPTS
C     DESIRED FILTER ORDER AND THE TEST FILTER COEFFICIENTS
C          ARE READ FROM A DATA FILE (FILE FT04F001).
          OPEN(UNIT=4,BLANK='ZERO')
C    INITIALIZE FILTER ORDER AND TEST FILTER COEFFICIENTS
          READ(4,100)N
          LL = N
          DO 20 K=0,LL
          READ(4,700)AR(K)
   20      CONTINUE
          NPTS = 4000
       WRITE(6,300)
       WRITE(6,400)N
          IX = 1913
       WRITE(6,600)NPTS
C     GENERATE (NPTS+1) WHITE NOISE SEQUENCE
          DO 10 I=0,NPTS
          CALL GAUSS(IX,1.,0.,V)
          X(I) = V
   10      CONTINUE
C    CREATE INPUT SEQUENCE FROM GIVEN FIR TEST FILTER
          CALL INPUT(LL,X,AR,Y,NPTS)
C    INITIALIZE AUTOCORRELATION VECTOR RX,RY, AND CROSSCORRELATION
C    VECTOR RXY
          CALL CORR(N+1,NPTS,X,Y,RX,RY,RXY)
          WRITE(6,800)
          DO 30 I=0,N
            WRITE(6,900)I,AR(I),RX(I)
   30     CONTINUE
C    INITIAL CONDITIONS FOR MOVING AVERAGE MODEL PARAMETERS
          B00 = -RXY(0)/RX(0)
          EY(0) = RY(0) - B00*RXY(0)
          EX(0) = RX(0)
          DO 40 I=0,1
          C(I) = I
          D(I) = I
   40     CONTINUE
          B(0) = 1.
          B(1) = B00
          DELY(0) = RXY(1) - B00*RX(1)
          DELX(0) = RX(1)
C    LOOP TO GENERATE PREDICTOR  COEFFICIENT VECTOR
          DO 120 M=1,N
C    INITIAL CONDITIONS FOR THE SYMMETRIC PREDICTOR POLYNOMIAL
C    CALCULATIONS.
          P(0,0) = 2.
          P(1,1) = 1.
          TAU(0) = RX(0)
          LAMDA(0) = 1.
          KODD = 1
```

```
      KEVEN = 2
      A(M,0) = 1.
C   SYMMETRIC SPLIT-LEVINSON RECURSION
      DO 70 K=1,M
      P(K,0) = 1.
C   CALL STATEMENTS FOR EVEN OR ODD VALUES OF INDEX K
      IF(K.EQ.KODD)THEN
         CALL ODD(RX,P,K,N,TAU,KODD,T)
        ELSEIF(K.EQ.KEVEN)THEN
         CALL EVEN(RX,P,K,N,TAU,KEVEN,T)
      ENDIF
      ALPHA(K) = TAU(K)/TAU(K-1)
C   LOOP TO CALCULATE SINGULAR PREDICTOR COEFFICIENTS
      DO 60 I=1,T
      P(K+1,I) = P(K,I) + P(K,I-1) - ALPHA(K)*P(K-1,I-1)
C   DECISION PATH TO CALCULATE SYMMETRIC SINGULAR PREDICTOR COEFFICIENTS
      IF(I.LT.T .OR. I.EQ.K)GOTO 60
        L = K+1
        DO 50 J=I+1,K
        P(L,J) = P(L,L-J)
50     CONTINUE
60     CONTINUE
      LAMDA(K) = 2. - (ALPHA(K)/LAMDA(K-1))
70     CONTINUE
C   CALCULATION OF N-TH ORDER PREDICTOR COEFFICIENTS, A(K,1),...A(K,K)
C      COMPUTE PREDICTOR COEFFICIENTS FOR K-TH ITERATION
      DO 80 I=1,M
       A(M,I) = A(M,I-1) + P(M+1,I) - LAMDA(M)*P(M,I-1)
       C(I) = A(M,I)
80     CONTINUE
      DO 90 J=1,M
       D(1) = -C(J)
       IF(J .LT. M)THEN
        DO 95 I=J+1,M
         D(I) = D(I-1)
95       CONTINUE
       ENDIF
90     CONTINUE
      D(0) = 0.
      D(M+1) = 1.
C   UPDATE PREDICTION ERRORS
      XBTMP = 0.
      XTMP = 0.
      DO 25 I=1,M
       XBTMP = XBTMP + C(I)*RXY(M+1-I)
       XTMP = XTMP + C(I)*RX(M+1-I)
25     CONTINUE
      DELX(M) = RX(M+1) - XTMP
      EXB(M) = RXY(M+1) - XBTMP
C   UPDATE REFLECTION COEFFICIENTS
```

```
          KX(M) = -DELX(M-1)/EX(M-1)
          EX(M) = EX(M-1) + KX(M)*DELX(M-1)
          KY(M) = -DELY(M-1)/EX(M)
          EY(M) = EY(M-1) + KY(M)*EXB(M)
C    UPDATE B VECTOR
          B(M+1) = 0.
          DO 45 I=0,M+1
           B(I) = B(I) + KY(M)*D(I)
   45     CONTINUE
          YTMP = 0.
          DO 55 I=1,M+1
          YTMP = YTMP - B(I)*RX(M+2-I)
   55     CONTINUE
          DELY(M) = RXY(M+1) - YTMP
          IF(M .EQ. N)THEN
           WRITE(6,1000)N
           WRITE(6,1100)
           DO 130 K=0,M
            WRITE(6,1200)K,-B(K+1)
            AA(K) = -B(K+1)
   130     CONTINUE
          ENDIF
   120    CONTINUE
   100    FORMAT(I2)
   200    FORMAT(' ')
   300    FORMAT('1')
   400    FORMAT('  FILTER ORDER = ',I3)
   500    FORMAT('-')
   600    FORMAT('-',' NUMBER OF SAMPLE POINTS  = ',I5)
   700    FORMAT(F8.4)
   800    FORMAT('-','5X,'INDEX',5X,'KNOWN COEFFICIENTS',5X,
     +'AUTOCORRELATION FUNCTION R(K)')
   900    FORMAT(' ',5X,I3,11X,F8.4,21X,F8.4)
  1000    FORMAT('-',10X,'PREDICTOR COEFFICIENTS FOR FILTER ORDER = ',I3)
  1100    FORMAT('-',5X,'INDEX',12X,'FIR PREDICTOR COEFFICIENTS')
  1200    FORMAT(' ',5X,I3,23X,F8.4)
       WRITE(6,300)
          END
       SUBROUTINE INPUT(LL,X,AR,Y,NPTS)
C      SUBROUTINE TO GENERATE THE INPUT SEQUENCE FROM A GIVEN FIR
C      FILTER AND ZERO MEAN, UNIT VARIANCE WHITE NOISE.
       REAL X(0:NPTS),AR(0:LL),Y(0:NPTS)
C    CALCULATE INPUT SEQUENCE VALUES BETWEEN 0 AND FILTER ORDER.
       DO 45 K=0,NPTS
       TEMP = 0.
       IF(K.LE.LL)THEN
        LU=K
       ELSE
        LU=LL
       ENDIF
```

```
      J = K
      DO 55 I=0,LU
      TEMP = TEMP + AR(I)*X(J)
      J = J-1
   55 CONTINUE
      Y(K) = TEMP
   45 CONTINUE
      RETURN
      END
      SUBROUTINE CORR(NLAG,NPTS,X,Y,RX,RY,RXY)
C     SUBROUTINE TO CALCULATE THE AUTOCORRELATION FUNCTION X AND Y AND
C     THE CROSSCORRELATION FUNCTION BETWEEN X AND Y
        REAL RX(0:NLAG),Y(0:NPTS),X(0:NPTS),RXY(0:NLAG),RY(0:2)
        INTEGER T
C   COMPUTE THE AUTOCORRELATION OF X AND THE CROSSCORRELATION OF
C   X AND Y FOR LAGS 0,1,2,...,NLAG
        DO 5 I=0,NLAG
        XTEMP = 0.
        XYTEMP = 0.
C   COMPUTE THE AUTOCORRELATION OF X AND THE CROSSCORRELATION OF
C   X & Y FOR LAG I
        DO 15 T=0,NPTS-1-I
         XTEMP = XTEMP + X(T)*X(T+I)
         XYTEMP = XYTEMP + X(T)*Y(T+I)
   15   CONTINUE
        RX(I) = XTEMP*(1./FLOAT(NPTS-1-I))
        RXY(I) = XYTEMP*(1./FLOAT(NPTS-1-I))
C   COMPUTE THE ZERO LAG AUTOCORRELATION FUNCTION OF Y
        IF(I .EQ. 0)THEN
         RY(0) = 0.
         DO 16 J=0,NPTS-1
          RY(0) = RY(0) + Y(J)**2
   16    CONTINUE
         RY(0) = RY(0)*(1./FLOAT(NPTS-1))
        ENDIF
    5   CONTINUE
        RETURN
        END
```

APPENDIX E. EXTENDED PRONY PROGRAM

```
C       PROGRAM TO CALCULATE THE NTH ORDER LATTICE REFLECTION
C       COEFFICIENTS FROM A GIVEN SEQUENCE USING THE SYMMETRIC ERROR
C       VECTOR, THE ASYMMTRIC ERROR VECTOR, OR THE FORWARD AND
C       BACKWARD ERROR VECTORS.
C             VARIABLE DEFINITIONS
C        PT   - TEMPORARY ARRAY USED TO AVERAGE PREDICTOR
C               COEFFICIENTS.
C        PP   - ESTIMATED NUMBER OF COMPLEX SINUSOIDS PRESENT.
C        A1   - AMPLITUDE OF #1 SINUSOID, (1-4) SINUSOIDS
C               PRESENT.
C        FS   - SAMPLING FREQUENCY.
C        F1   - FREQUENCY OF #1 SINUSOID IN TEST SEQUENCE.
C     THETA1  - DIGITAL FREQUENCY OF #1 TEST ANALOG FREQUENCY.
C              VARIABLE DECLARATIONS
      REAL W(0:5000),S(0:5000),ALPHA(100),ROOTR(100),XCOF(0:100)
      REAL P(0:100,0:100),ALPHAS(100),COF(0:100),ROOTI(100)
      REAL X1(0:5000),XO(0:5000),AT(0:5000),PS(0:100,0:100),PT(0:100)
      INTEGER T,PP
         OPEN(UNIT=4,BLANK='ZERO')
C   INITIALIZE FILTER ORDER
         READ(4,100)PP
         M = 2*PP
C   INITIAL CONDITIONS FOR INPUT SEQUENCE GENERATOR
         IX = 1913
         A = SQRT(2.)
         B = SQRT(2.)
         C = SQRT(10.)
C        D = SQRT(2.0)
C        E = SQRT(2.0)
         F1 = 5.5E+01
         F2 = 7.5E+02
C        F3 = 1.25E+02
C        F4 = 1.75E+02
         FS = 2.25E+02
         TWOPI = 6.2831853
         THETA1 = (TWOPI*F1)/FS
         THETA2 = (TWOPI*F2)/FS
C   LOOP TO GENERATE WHITE NOISE SEQUENCE AND TO READ TEST COEFFICIENT
         DO 1 I=0,N
         CALL GAUSS(IX,1.,0.,V)
         W(I) = C*V
         A1 = A*COS(TWOPI*(F1/FS)*FLOAT(I))
         A2 = B*COS(TWOPI*(F2/FS)*FLOAT(I))
C        A3 = D*COS(TWOPI*(F3/FS)*FLOAT(I))
C        A4 = E*COS(TWOPI*(F4/FS)*FLOAT(I))
         S(I) = A1 + A2 + W(I)
```

```
  1      CONTINUE
C  CALL STATEMENTS FOR SYMMETRIC, ASYMMETRIC, AND LEVINSON LATTICE
C  RECURSION SUBROUTINES
       CALL SLAT(S,M,N,P,ALPHA,X1,AT,XO)
C      CALL ASLAT(S,M,N,PS,ALPHAS,X1,AT,XO)
     WRITE(6,200)
       WRITE(6,300)PP
     WRITE(6,400)M
     WRITE(6,500)N
C      DISPLAY COEFFICIENTS OF POLYNOMIAL
       WRITE(6,600)
       DO 11 K=0,M
       IF(K .EQ. M)P(M,K)=1.0
       WRITE(6,700)M,K,P(M,K)
 11    CONTINUE
 100   FORMAT(I4)
 200   FORMAT('1')
 300   FORMAT(' NUMBER OF COMPLEX EXPONENTIALS IN MODEL = ',I3)
 400   FORMAT(' ',' SYMMETRIC FILTER ORDER = ',I3)
 500   FORMAT(' ',' NUMBER OF SAMPLE POINTS = ',I5)
 600   FORMAT('-',8X,'INDEX',13X,'COEFFICIENTS')
 700   FORMAT(5X,2I2,16X,F8.4)
     WRITE(6,200)
       END
      SUBROUTINE SLAT(S,M,N,P,ALPHA,X1,AT,XO)
C      SUBROUTINE TO COMPUTE THE SYMMETRIC PREDICTOR COEFFICIENTS
C      USING THE SYMMETRIC PREDICTION ERROR VECTOR.
       REAL X1(0:M+N),XO(0:M+N),ALPHA(M),S(0:N)
       REAL AT(0:M+N),P(0:100,0:100)
       INTEGER T
C      INITIAL CONDITIONS
       KODD = 1
       KEVEN = 2
       TAU = 0.
       P(1,1) = 1.
       P(0,0) = 2.
C   INITIALIZE THE PREDICTION ERROR VECTORS FOR THE ZERO AND 1ST
C   STAGES OF THE LATTICE.  INITIALIZE THE ZERO STAGE MODIFIED NORM
       DO 10 T=0,N-1
       XO(T) = 2.*S(T)
       TAU = TAU + S(T)**2
 10    CONTINUE
       DO 20 T=0,N
        IF(T.EQ.N)S(T) = 0.
        IF(T.EQ.0)THEN
         X1(T) = S(T)
        ELSE
         X1(T) = S(T) + S(T-1)
        ENDIF
```

```
  20      CONTINUE
C     LOOP TO COMPUTE THE SYMMETRIC PREDICTOR COEFFICIENTS
        DO 101 K=1,M
        P(K,0) = 1.0
        TTAU = TAU
         IF(K .EQ. KODD)THEN
          LL = (K+1)/2
         ELSE
          LL = K/2
         ENDIF
C      STORE TAU(K-1), AND COMPUTE TAU(K).
        TAU = 0.
        DO 30 T=0,N+K-1
         TAU = TAU + X1(T)**2
  30    CONTINUE
        TAU = TAU/2.
C      COMPUTE ALPHA(K); STORE TAU(K) FOR NEXT ITERATION.
        ALPHA(K) = TAU/TTAU
        TTAU = TAU
C    LOOP TO CALCULATE SINGULAR PREDICTOR COEFFICIENTS
        DO 40 I=1,LL
        P(K+1,I) = P(K,I) + P(K,I-1) - ALPHA(K)*P(K-1,I-1)
C    DECISION PATH TO CALCULATE SYMMETRIC SINGULAR PREDICTOR COEFFICIENTS
        IF(I.LT.LL .OR. I.EQ.K)GOTO 40
          L = K+1
          DO 50 J=I+1,K
          P(L,J) = P(L,L-J)
  50      CONTINUE
  40    CONTINUE
C      LOOP TO COMPUTE THE (K+1) PREDICTION ERROR VECTOR.
        DO 60 T=0,N+K
         IF(T .EQ. 0)THEN
          AT(T) = X1(T)
         ELSEIF(T.EQ.N+K)THEN
          AT(T) = X1(T-1)
          ELSE
           AT(T) = X1(T) + X1(T-1) - ALPHA(K)*XO(T-1)
          ENDIF
  60    CONTINUE
C      LOOPS TO UPDATE PREDICTION ERROR VECTORS FOR NEXT ITERATION.
C         (SHIFT X1 INTO XO AND AT INTO X1)
        DO 70 T=0,N+K-1
         XO(T) = X1(T)
  70    CONTINUE
        DO 80 T=0,N+K
         X1(T) = AT(T)
  80    CONTINUE
        IF(K .EQ. KODD)THEN
         KODD = KODD + 2
        ELSE
```

```
          KEVEN = KEVEN + 2
         ENDIF
  101    CONTINUE
         RETURN
         END
         SUBROUTINE ASLAT(S,M,N,PS,ALPHAS,X1,AT,XO)
C        SUBROUTINE TO COMPUTE THE LATTICE REFLECTION COEFFICIENTS
C        USING THE ASYMMETRIC PREDICTION ERROR VECTOR.
         REAL X1(0:M+N),XO(0:M+N),PS(0:100,0:100),S(0:N),ALPHAS(M)
         REAL AT(0:M+N)
         INTEGER T
C        INITIAL CONDITIONS
         PS(0,0) = 2.
         PS(1,1) = 1.
         KODD = 1
         KEVEN = 2
         TAU = 0.
C   INITIALIZE THE PREDICTION ERROR VECTORS FOR THE ZERO AND 1ST
C   STAGES OF THE LATTICE. INITIALIZE THE ZERO STAGE MODIFIED NORM
         DO 10 T=0,N-1
          XO(T) = 0.
          TAU = TAU + S(T)**2
  10     CONTINUE
         DO 20 T=0,N
          IF(T.EQ.N)X1(T) = -S(T-1)
          IF(T.EQ.0)THEN
           X1(T) = S(T)
          ELSE
           X1(T) = S(T) - S(T-1)
          ENDIF
  20     CONTINUE
C    LOOP TO COMPUTE THE REFLECTION COEFFICIENTS
         DO 101 K=1,M
         PS(K,0) = 1.
          TTAU = TAU
         IF(K .EQ. KODD)THEN
          LL = (K-1)/2
         ELSE
          LL = K/2
         ENDIF
C     STORE TAU(K-1), AND COMPUTE TAU(K).
          TAU = 0.
         DO 30 T=0,N+K-1
          TAU = TAU + X1(T)**2
  30     CONTINUE
         TAU = TAU/2.
C     COMPUTE ALPHA(K) ; STORE TAU(K) FOR NEXT ITERATION.
         ALPHAS(K) = TAU/TTAU
         TTAU = TAU
C   LOOP TO CALCULATE SINGULAR PREDICTOR COEFFICIENTS
```

```
          DO 40 I=1,LL
          PS(K+1,I) = PS(K,I) + PS(K,I-1) - ALPHAS(K)*PS(K-1,I-1)
C    DECISION PATH TO CALCULATE SYMMETRIC SINGULAR PREDICTOR COEFFICIENTS
          IF(I.LT.LL .OR. I.EQ.K)GOTO 40
            L = K+1
            DO 50 J=I+1,K
            PS(L,J) = -PS(L,L-J)
   50       CONTINUE
   40     CONTINUE
C      LOOP TO COMPUTE THE (K+1) PREDICTION ERROR VECTOR.
          DO 60 T=0,N+K
           IF(T .EQ. 0)THEN
            AT(T) = X1(T)
           ELSEIF(T.EQ.N+K)THEN
            AT(T) = X1(T-1)
            ELSE
             AT(T) = X1(T) + X1(T-1) - ALPHAS(K)*XO(T-1)
            ENDIF
   60     CONTINUE
C      LOOPS TO UPDATE PREDICTION ERROR VECTORS FOR NEXT ITERATION.
C         (SHIFT X1 INTO XO AND AT INTO X1)
          DO 70 T=0,N+K-1
           XO(T) = X1(T)
   70     CONTINUE
          DO 80 T=0,N+K
           X1(T) = AT(T)
   80     CONTINUE
          IF(K .EQ. KODD)THEN
           KODD = KODD + 2
          ELSE
           KEVEN = KEVEN + 2
          ENDIF
  101     CONTINUE
          RETURN
          END
```

LIST OF REFERENCES

1. Delsarte, Phillipe and Genin, Yves V., *The Split Levinson Algorithm*, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-35 No. 5, May, 1987

2. Delsarte, Phillipe and Genin, Yves V., *On the Splitting of Classical Algorithms in Linear Prediction Theory*, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-35 No. 6, June, 1987

3. Orfanidis, Sophocles J., *Optimum Signal Processing An Introduction*, McMillan Publishing Company, New York, 1985

4. Haykin, Simon, *Introduction to Adaptive Filters*, McMillan Publishing Company, New York, 1984

5. Martinelli, G., Orlandi, G., and Burrascano, P., *Yule-Walker Equations and Barlett's Bisection Theory*, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. CAS-32 No. 10, October, 1985

6. Tummala, Murali, " Least squares identification of moving average models using linear prediction techniques and iterative algorithms," ECE Dept. Technical Report (under preparation) , Naval Postgraduate School, Monterey, CA 93943

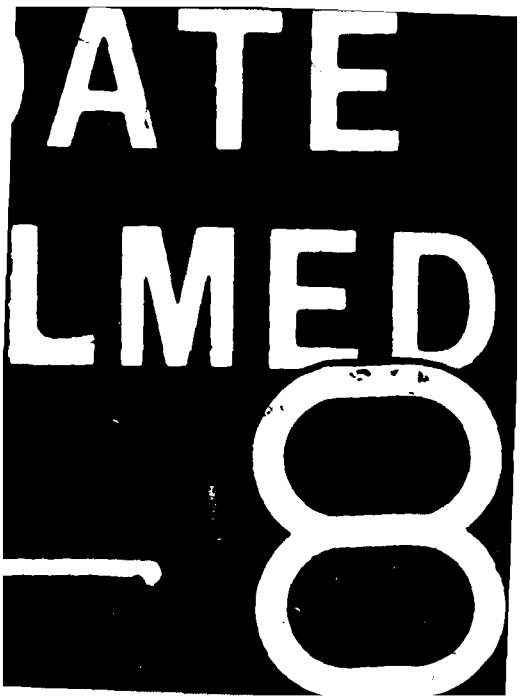
7. Kay, Steven M., and Marple, Stanley L., Jr., *Spectrum Analysis - A Modern Prespective*, Proceedings of the IEEE Vol. 69, No. 11, Nov. 1981

8. Kumaresan, R., and Tufts, D.W. *Improved Spectral Resolution III - Efficient Realization*, Proceeding of the IEEE, Vol. 68, No. 10, Oct. 1980, pp. 1354-1355

9. Kumaresan, R., Tufts, D.W., and Scharf, L.L *A Prony Meyhod for Noisy Data - Choosing the Signal Components and Selecting the Order in Exponential Signal Models*, Proceeding of the IEEE, Vol. 72, No. 2, Feb. 1984, pp. 230-233

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman, Code 62 Electrical and Computer Engineering Department Naval Postgraduate School Monterey, CA 93940-5000	1
4.	Dr. Murali Tummala Electrical and Computer Engineering Department Naval Postgraduate School Monterey CA 93940-5000	2
5.	Dr. Ralph D. Hippenstiel Electrical and Computer Engineering Department Naval Postgraduate School Monterey, CA 93940-5000	1
6.	Commander Naval Ocean Systems Command ATTN: Dr. Eugene P. Cooper, Code 013 San Diego, CA 92152-5000	1
7.	Supervisor of Shipbuilding Conversion & Repair, USN ATTN: Ledr William A. Dicken New Orleans, LA 70142-5093	2


ATE
LMED
8